Algorithmen im Wirkstoffdesign

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Herrn Dipl.-Math. Martin Thimm
geboren am 07.07.1970 in Schramberg

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Christoph Markschies

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. Uwe Küchler

Gutachter:

1. Prof. Dr. Hans Jürgen Prömel
2. Priv.-Doz. Dr. Stefan Hougardy
3. Prof. Dr. Cornelius Frömmel

eingereicht am:	25. August 2005
Tag der mündlichen Prüfung:	17. Januar 2006

Abstract

Two important questions in drug design are the following: "How to compute the similarity of two molecules?" and "How to cluster molecules by similarity?"

In the first part we describe two different approaches to compare molecules for 3D-similarity.

The first algorithm just uses the 3D coordinates of the atoms as input. We show that this algorithm is able to detect similar activity or similar adverse reaction, even with a simple purely geometry based scoring function. Compared to previous simpler approaches more interesting hits are found.

The connectivity structures of the molecular graphs are used by the second algorithm as additional input. This fully flexible approach – conformers of the molecules are treated simultaneously – may even find provably optimal solutions. Parameter settings for practically relevant instances allow running times that make it possible to even search large databases.

The second part describes two methods to search a database of molecular structures. After analyzing the data with graph theoretical methods two algorithms for two different ranges of similarity are designed. Scanning the database for structures similar to a given query can be accelerated considerably. We use hierarchical methods and dominating set techniques.

Finally we propose a new biclustering algorithm. Biclustering problems recently appeared mainly in the context of analysing gene expression data. Again graph theoretical methods are our main tools. In our model biclusters correspond to dense subgraphs of certain bipartite graphs. In a first phase the algorithm deterministically finds supersets of solution candidates. Thinning out these sets by heuristical methods leads to solutions. This new algorithm outperforms former comparable methods and its simple structure make it easy to customize it for practical applications.

Keywords:
drug design, algorithm, 3D superposition, Branch and Bound

Zusammenfassung

Die Bestimmung der Ähnlichkeit von molekularen Strukturen und das Clustern solcher Strukturen gemäß Ähnlichkeit sind zwei zentrale Fragen im Wirkstoffdesign.

Die Arbeit beschreibt im ersten Teil zwei neue Verfahren zum Vergleich von Molekülen auf 3-dimensionale Ähnlichkeit.

Der erste Algorithmus benutzt als Eingabe nur die Koordinaten der Atome der zu vergleichenden Moleküle. Wir können zeigen, daß eine rein geometrische Zielfunktion in der Lage ist, Wirkungsähnlichkeit von Substanzen vorherzusagen, und daß der Algorithmus geeignet ist, Ähnlichkeiten zu finden, die mit bisherigen, einfacheren Methoden nicht gefunden werden konnten.

Das zweite Verfahren nutzt zusätzlich noch die Bindungsstruktur der Eingabemoleküle. Es ist flexibel, d.h. alle Konformere der Moleküle werden simultan behandelt. Wir erhalten ein sehr schnelles Verfahren, das bei geeigneter Parametereinstellung auch beweisbar optimale Lösungen liefert. Für praktisch relevante Anwendungen erreichen wir erstmals Laufzeiten, die selbst das Durchsuchen großer Datenbanken ermöglichen.

Im zweiten Teil beschreiben wir zwei Methoden, eine Menge von molekularen Strukturen so zu organisieren, daß die Suche nach geometrisch ähnlichen deutlich schneller durchgeführt werden kann als durch lineare Suche. Nach Analyse der Daten mit graphentheoretischen Methoden finden hierarchische Verfahren und repräsentantenbasierte Ansätze ihre Anwendung.

Schließlich geben wir einen neuen Algorithmus zum Biclustern von Daten an, einem Problem, das bei der Analyse von Genexpressionsdaten eine wichtige Rolle spielt. Mit graphentheoretischen Methoden konstruieren wir zunächst deterministisch Obermengen von Lösungen, die danach heuristisch ausgedünnt werden. Wir können zeigen, daß dieser neue Ansatz bisherige, vergleichbare z.T. deutlich überbietet. Seine prinzipielle Einfachheit läßt anwendungsbezogene Modifikationen leicht zu.

Schlagwörter:
Wirkstoffdesign, Algorithmus, 3D-Überlagerung, Branch and Bound

Inhaltsverzeichnis

Kapitel 1

Einleitung

In dieser Arbeit beschäftigen wir uns mit zwei grundlegenden und wichtigen Fragen im Wirkstoffdesign: „Wie ähnlich sind sich zwei Moleküle?“ und „Wie kann eine große Menge von Molekülen in Klassen von jeweils sehr ähnlichen unterteilt werden?“

Die Allgemeinheit, in der diese Fragen hier formuliert werden, soll nicht andeuten, daß wir sie in dieser allgemeinen Form auch lösen können, vielmehr soll sie helfen, die behandelten Fragestellungen in einen größeren Zusammenhang einzuordnen.

Aus Sicht der Anwendung, dem Wirkstoffdesign, folgen wir der allgemein akzeptierten Annahme, daß ähnliche Wirkung (oder Nebenwirkung) von Substanzen in engem Zusammenhang steht mit ähnlicher molekularer Struktur. Die Definition – was unter ähnlicher molekularer Struktur zu verstehen ist – muß weiter präzisiert werden, genauso die Frage, wie diese dann gemessen und bewertet wird.

Die Unterteilung der jeweils betrachteten Gesamtmenge von Strukturen in Klassen von zueinander ähnlichen hängt in entscheidender Weise von der Art der Ähnlichkeitsdefinition ab. Erst danach kann über geeignete Verfahren zum Bestimmen dieser Klassen entschieden werden.

Hauptanliegen dieser Arbeit ist es, auf der Ebene der Methoden neue Beiträge zu liefern, weniger im konkreten Suchen nach neuen Wirkstoffen oder Wirkungszusammenhängen. Wir konzentrieren uns im Wesentlichen auf effektive Methoden zur Bestimmung der 3–dimensionalen Ähnlichkeit von Molekülen und auf Verfahren, diese gemäß einer solchen Ähnlichkeit geeignet zu gruppieren und zu sortieren.

Die Darstellung der Resultate ist aus diesem Grund in die Kapitel „Algorithmen“ (2) und „Anwendungen“ (3) geteilt. Im ersten Teil beschreiben wir die neu entwickelten Verfahren mehr aus methodischer und algorithmischer Sicht, im zweiten zeigen wir anhand konkreter Anwendungen, wie sich diese

neuen Methoden gewinnbringend einsetzen lassen und neue, bisher nicht zu erreichende Resultate liefern.

Wirkstoffdesign

Sollen Medikamente gezielt gefunden oder sogar entworfen werden, so müssen wir zuerst die folgende sehr einfache Frage beantworten: Wie wirkt überhaupt ein Medikament? Schon vor mehr also 100 Jahren hat Emil Fischer das Bild von „Schlüssel und Schloß“ geprägt, das bis heute Geltung hat: Der Ligand paßt an oder in den Rezeptor wie ein passender Schlüssel in ein Schloß (siehe z.B. [BKK02]). D.h also, daß eine notwendige Voraussetzung dafür – neben einigen anderen – daß ein Molekül an ein Protein bindet und dadurch eine Wirkung erzielt, die passende 3–dimensionale Struktur von Molekül und entsprechender Bindungsstelle am Protein ist. Hier findet sich der Hauptuntersuchungsgegenstand vorliegender Arbeit: Vergleich von Molekülen auf 3-dimensionale Ähnlichkeit. Andere Voraussetzungen und Gründe für die Protein–Ligand–Wechselwirkung werden für unsere Problemstellung weniger wichtig sein. Auch weitere im Wirkstoffdesign wichtige Fragestellungen werden hier nicht weiter betrachtet, z.B. Toxizität oder Pharmakokinetik, also alle Fragen, die sich mit der Aufnahme, Verteilung, Metabolismus und Ausscheidung der zugeführten Substanz im Körper beschäftigen.

Wir können zwei grundlegend unterschiedliche Szenarien vorfinden: Ist die exakte 3–dimensionale Struktur der Bindungsstelle bekannt oder nicht? Anders formuliert: Kennen wir das Schloss und suchen einen passenden Schlüssel? Im anderen Fall kennen wir vielleicht nur einen oder mehrere passende Schlüssel und versuchen durch Vergleichen mit diesen weitere, eventuell noch passendere Schlüssel zu finden [PGF00].

Im ersten Fall sind wir mit dem sogenannten *docking–Problem* konfrontiert: Die exakte 3–dimensionale Struktur der Bindungsstelle am Zielprotein ist bekannt. Die Aufgabe besteht dann darin, eine gegebene Menge von potentiellen Liganden daraufhin zu testen und zu bewerten, wie gut sie an dieser Bindungsstelle andocken können und wie der entstehende Protein–Ligand–Komplex aussieht. Neben auch hier wichtigen geometrischen Bedingungen spielen noch andere, vor allem physikalische Bedingungen, eine entscheidende Rolle: Die freie Energie des potentiellen Komplexes sollte so niedrig wie möglich sein. Der Einfluß von Wasser als umgebendes Medium stellt ein eigenes, noch nicht befriedigend gelöstes Problem dar. Dieser Themenkomplex ist Gegenstand lebendiger Forschung und soll hier nicht weiter ausgeführt werden. Einen guten Einblick in die Thematik und einen Überblick über aktuelle, relevante Literatur liefert [SSK03].

Noch detaillierter mit dem potentiellen Protein–Ligand–Komplex, sind Bindungsstelle und vielleicht einige aussichtsreiche Liganden–Kandidaten bekannt, befaßt sich das sogenannte *strukturbasierte Wirkstoffdesign* [And03]. Nachdem die potentiellen und aussichtsreichsten Bindungsstellen am Protein identifiziert sind und erste Liganden durch Screening oder vorheriges Wissen vorliegen, werden diese getestet und der so entstehende Komplex erneut strukturell untersucht. Die auf diese Weise erhaltenen Kenntnisse ermöglichen es danach, in einem weiteren Zyklus, den Liganden gezielt zu verändern, um bessere Bindungseigenschaften zu erhalten.

Der direkteste Ansatz, das *De Novo–Design*, versucht nun, ohne Kenntnis einer Leitstruktur, nur durch Untersuchung der Bindungsstelle des Proteins, einen Liganden sozusagen „am Reißbrett“ zu entwerfen (siehe z.B. [STJ+02]). Eine Vorgehensweise ist beispielsweise, einen Liganden dadurch zu erstellen, daß kleine Stücke von Molekülen zuerst einzeln in die Bindungstasche des Proteins eingepaßt werden, die schließlich zu einem Zielmolekül zusammengebaut werden.

Betrachten wir nun das zweite Szenario: Wir kennen die Bindungsstelle des Proteins nicht und nur einige wenige potentielle Liganden sind bekannt, vielleicht teilweise mit ihrer gemessenen Aktivität. Mit Hilfe indirekter Methoden wird nun einzig auf Basis der Liganden–Daten weitergearbeitet.

Ausgangspunkt ist der Versuch, durch $3D$-Überlagerung der Liganden die für den Bindungsprozess relevanten Informationen zu gewinnen. An dieser Stelle setzt unsere Arbeit an.

Finden sich starke Überlagerungsübereinstimmungen, so kann zum einen versucht werden, die für den Bindungsprozess relevanten funktionalen Gruppen (in ihrer räumlichen Lage), den sogenannten *Pharmakophor*, zu extrahieren. Ist ein solcher Pharmakophor bekannt, kann er in einer weiteren Iteration erneut benutzt werden, um neue potentielle Liganden zu identifizieren (siehe z.B. [KG01]).

In sogenannten *QSAR*–Studien kann zum anderen der Versuch erfolgen, $3D$–Information mit in das Modell aufzunehmen. Quantitative Struktur–Wirkungsbeziehung (QSAR, von engl. quantitative structure activity relationship) betrachtet die Zusammenhänge zwischen chemischen Strukturen und biologischer Wirkung und erfaßt und beschreibt diese quantitativ [Kub03]. Wir wollen hier nicht näher auf diesen eigenständigen Bereich der Forschung eingehen, jedoch bemerken, daß zu den chemischen Eigenschaften, die mit biologischer Aktivität zu korrelieren man versucht, auch solche gehören können, die die $3D$-Struktur der betrachteten Moleküle betreffen. Man spricht dann auch von $3D$-QSAR.

Ein weiterer zentraler Punkt ist die Flexibilität der betrachteten Strukturen. Sowohl Protein wie auch Ligand sind nicht starr, sondern können ver-

schiedene, energetisch günstige räumliche Formen annehmen. Diese werden *Konformere* genannt. Die Konformationsanalyse, also die Untersuchung, welche dieser Formen auftreten können, und wie diese effektiv erzeugt werden, ist ein eigenes Forschungsgebiet. Einen guten Überblick liefert z.B. [Sch03].

Zentrales Anliegen der vorliegenden Arbeit ist es, Verfahren zum effektiven Vergleich von Molekülen und molekularen Strukturen auf 3-dimensionale Ähnlichkeit bereitzustellen. Der Begriff der Ähnlichkeit muß hierzu genauer definiert werden. Dies, wie auch ein Überblick über die in der Literatur zu findenden Ansätze zur Lösung dieser Aufgabe, findet sich in einem eigenen Abschnitt (siehe 2.1.1).

Clustern

Wenden wir uns der zweiten zu Beginn dieser Einleitung gestellten Frage zu: „Wie kann eine große Menge von Molekülen in Klassen von jeweils sehr ähnlichen unterteilt werden?“ Haben wir nun mit den noch zu beschreibenden Verfahren molekulare Strukturen auf Ähnlichkeit untersucht, so ist es oft wichtig, Klassen von paarweise sehr ähnlichen Elementen zu bilden oder solche Klassen von Elementen zu unterscheiden, die sich sehr unähnlich sind. Ganz allgemein kann dieser Prozess als „Clustern“ bezeichnet werden.

Es würde den Rahmen dieser Arbeit bei weitem sprengen, einen allgemeinen und erschöpfenden Überblick über dieses Gebiet zu geben. Die Anzahl der verschiedenen Ansätze ist so zahlreich wie die verschiedenen Anwendungsgebiete.

Wir zeigen exemplarisch, daß in einem Fall eine genaue Analyse der Eingabedaten helfen kann, maßgeschneiderte Clusteralgorithmen zu entwickeln, die den bekannten Standardansätzen dann weit überlegen sind (2.2.2). In einem anderen Fall (2.2.3) legen wir dar, daß es sinnvoll und hilfreich sein kann, zumeist benutzte Ähnlichkeitsmaße zu überprüfen und gegebenenfalls leicht abzuändern, um danach klarere Aussagen zu erhalten.

Schließlich betrachten wir ein Verfahren zum Biclustern von Daten. Probleme dieses Typs treten in den letzten Jahren vermehrt im Zusammenhang mit der Analyse von Genexpressionsdaten auf: Gegeben sind eine Menge von Objekten und eine Menge von Bedingungen, unter denen alle diese Objekte untersucht wurden. Ziel ist es nun, Teilmengen von Objekten zu identifizieren, die sich unter einer Teilmenge von Bedingungen sehr ähnlich verhalten. Eine genauere Definition des von uns betrachteten Problems, bekannte Ansätze und Zielfunktionen, beschreiben wir in Abschnitt 2.2.1.

Unsere Beiträge

$3D$-Ähnlichkeit

In der Terminologie der mathematischen Optimierung bezeichnen wir mit *Überlagerungsproblem* die abstrakte Aufgabe der Optimierung der Score–Wertes (siehe Definition 2.1.5) für beliebige, zulässige Eingaben. Eine zulässige Eingabe – zwei konkrete Moleküle, die es zu überlagern gilt – nennen wir eine *Instanz* des Überlagerungsproblems. Wie sagen, daß wir das Überlagerungsproblem *exakt* lösen können, wenn ein Algorithmus existiert, der für alle möglichen Instanzen des Problems die beweisbar optimale Lösung berechnet. Von einer *Näherungslösung* bzw. *Approximation* sprechen wir, wenn die Ausgabe des Algorithmus – also eine konkrete Überlagerungszuordnung mit zugehörigem Score–Wert – nicht notwendig optimal ist.

Ein Verfahren zum Vergleich molekularer Strukturen, das als Eingabeinformation lediglich die $3D$–Koordinaten der Atome der zu vergleichenden Moleküle benutzt, wird in Abschnitt 2.1.3 beschrieben.

Wir benutzen hier als Unterroutine einen exakten Algorithmus (siehe 2.1.2.2) zur Lösung des Überlagerungsproblems, wie es in 2.1.2.1 genau beschrieben wird. Dieses als *Branch and Bound*–Algorithmus implementierte Verfahren ist für sich allein in der Lage, Instanzen mit bis zu 16 Atomen in angemessener Zeit exakt zu lösen, was nur durch den Beweis und die Anwendung von effektiven Schranken und durch das Bestimmen geeigneter Verzweigungsreihenfolgen erreichbar ist.

Um nun aber größere Instanzen schneller bearbeiten zu können, gehen wir wie folgt vor: Die Originaleingabe wird in einer Weise vergröbert, daß einerseits die Information über die räumliche Gestalt der Ausgangsstrukturen im Wesentlichen erhalten bleibt, jedoch andererseits die so erhaltene neue Struktur als ein kleines Molekül erscheint. Diese neuen, kleinen „Pseudomoleküle“ können dann mit dem exakten Algorithmus schnell überlagert werden. Schließlich heben wir diese Zwischenlösung geeignet auf die Originalinstanzen hoch, d.h. wir verfeinern die Lösung auf den „Pseudomolekülen“, indem wir die im ersten Schritt vorgenommene Vergröberung wieder rückgängig machen, um so eine Lösung für die Ausgangsmoleküle zu erhalten. Eine Überlagerungszuordnung für zwei „Pseudoatome“ wird also zu einer Menge von Zuordnungen für die in den „Pseudoatomen“ zusammengefaßten Ausgangsatome.

In Abschnitt 3.1 zeigen wir, daß dieser Algorithmus Ähnlichkeit zwischen bestimmten Oberflächenstrukturen, wie sie in [PGF98] beschrieben wurden, wesentlich besser berechnen kann als das dort angewandte Verfahren; allerdings auch mit erheblich höherer Laufzeit.

In einer zweiten wichtigen Anwendung können wir sehen, daß dieser Algorithmus in einer Datenbanksuche mit Medikamentenwirkstoffen in der Lage ist, allein aufgrund der Geometrie der Eingabedaten, nahezu alle wichtigen Treffer zu finden (3.2). Diese Ergebnisse sind in Zusammenarbeit mit Robert Preissner, Andrean Goede und Stefan Hougardy entstanden [TGHP04]. Der Algorithmus ist in Form eines Web–Interfaces öffentlich zugänglich [HRT04].

Ein noch wesentlich effektiveres Verfahren zum $3D$–Ähnlichkeitsvergleich wird in Abschnitt 2.1.4 beschrieben. Wir benutzen hier zusätzlich zu den Koordinaten der Atome auch die topologische Struktur des Bindungsgraphen, jedoch keine weiteren chemischen oder physikalischen Eigenschaften der Moleküle. Ein Kernpunkt des Verfahrens ist es, daß wir beim Vergleich zweier Moleküle simultan alle Konformere betrachten können, und diese nicht wie bei starren Überlagerungsverfahren (wie auch in 2.1.3), paarweise vergleichen müssen. Die Zuhilfenahme der Bindungsstrukturen liefert mit einem effektiven direkten Algorithmus zur Enumeration relevanter Substrukturen den Grundstein für ein Branch and Bound–Verfahren. Mehrere effektive Schranken, durch Berücksichtigung der Lage der Atome über alle Konformere, machen es möglich, je nach Parameterwahl, sogar exakte Lösungen des Überlagerungsproblems für relevante Treffer sicherzustellen. Das Verfahren ist also in der Lage, die bestmöglichen Überlagerungen zu berechnen, wenn wir nur nach solchen Molekülen suchen, die sich sehr ähnlich sind.

Wir können zeigen, daß die Laufzeit des Verfahrens, im Vergleich zu dem in (2.1.3) etwa um den Faktor $500-1000$ besser ist, die Qualität der Resultate aber mindestens gleichbleibt.

In einem zweiten Anwendungsbeispiel (3.3) zeigen wir, daß der Algorithmus tatsächlich in der Lage ist, große Datenmengen zu behandeln.

Wir sehen dort auch, daß Information über $3D$–Ähnlichkeit, gepaart mit anderen Ähnlichkeitsmaßen (z.B. Tanimotokoeffizienten), helfen kann, Wirkungszusammenhänge besser aufzufinden.

Das in 2.1.4 beschriebene Verfahren, entstanden in Zusammenarbeit mit Stefan Hougardy und Valentin Ziegler, ist Inhalt der deutschen Patentanmeldung [HTZ05].

Clustern

Wir zeigen in Abschnitt 2.2.2, wie es möglich ist, die Datenbank DIP [PGF98] in sublinearer Zeit nach ähnlichen molekularen Strukturen zu durchsuchen. Hilfreich sind hier eine eingehende Analyse der Struktur der Daten wie auch darauf speziell zugeschnittene Cluster– und Suchalgorithmen. Je nach Grad der gesuchten Ähnlichkeit können wir *brute–force*–Ansätze in der Laufzeit um den Faktor 25 (für sehr hohe Ähnlichkeit) und um einen Faktor nahe 10

(für mittlere Ähnlichkeit) übertreffen. Diese Resultate ergaben sich in Zusammenarbeit mit Cornelius Frömmel, Christoph Gille, Andrean Goede, Clemens Gröpl, Stefan Hougardy, Till Nierhoff und Robert Preissner [FGG+03].

Der in 2.2.4 beschriebene Algorithmus zum Biclustern von Daten – eine Kombination von Verfahren zur gezielten Einschränkung des Lösungsraums und heuristischen Methoden zum Auffinden von dichten Unterstrukturen, entstanden in Zusammenarbeit mit Valentin Ziegler, – liefert Ergebnisse, die denen in [CC00] und [YWWY02] z.T. deutlich überlegen sind (3.4.2).

Kapitel 2

Algorithmen

2.1 Strukturvergleich

Ähnlichkeit molekularer Strukturen kann auf mannigfaltige Weise definiert und berechnet werden. Verschiedene Sichtweisen und Anwendungen bedingen verschiedene Ansätze. Jeder Ansatz ist für sich meist gerechtfertigt durch ein einerseits plausibles Modell, andererseits durch Resultate, die auf andere Weise vorher nicht erreicht werden konnten. Die Vielfalt der Ähnlichkeitsdefinitionen und der darauf operierenden Such– und Vergleichsalgorithmen macht es sehr schwer, einzelne Ansätze direkt zu vergleichen, genauso wie die Diversität der behandelten Daten.

Ein vollständiger Überblick über alle relevanten, in der Literatur besprochenen Verfahren, würde aufgrund der Fülle den Rahmen dieser Arbeit sprengen. Im Folgenden wollen wir versuchen, verschiedene Hauptansätze prinzipiell zu skizzieren. Wir beschränken uns auf die Gebiete, die zumindest ansatzweise Ähnlichkeit zu unseren Vorgehensweisen haben, also vornehmlich algorithmisch orientiert sind und die topologische oder räumliche Struktur der zugrunde liegenden Objekte mitberücksichtigen.

2.1.1 Überblick

2.1.1.1 Substruktursuche

Die Substruktursuche ist die einfachste und historisch älteste Art, molekulare Strukturen zu vergleichen. Ziel ist es, aus einer Menge von Strukturen all jene herauszufiltern, die eine vordefinierte Substruktur genau enthalten. Die Definition der zu findenden Substruktur ist Sache des Anwenders.

In den allermeisten Anwendungen werden Moleküle (und ihre Substrukturen) als Graphen (und ihre Untergraphen) modelliert. Zu einem gegebenen

Molekül M konstruiert man den dazugehörigen Graphen G_M auf die folgende Weise: Die Atome von M entsprechen den Knoten von G_M, die Bindungen in M den Kanten von G_M. Meist tragen die Knoten von G_M Bezeichnungen oder Labels, die die chemischen Eigenschaften der Atome beschreiben (z.B. Atomtyp); genauso beschreiben die Bezeichnungen an den Kanten von G_M chemische Eigenschaften der Bindungen (z.B. Bindungstyp).

Suchen wir nun eine gegebene Struktur S in einem Molekül M, so ist dies gleichbedeutend mit der Frage, ob G_M einen zu G_S isomorphen induzierten Untergraphen besitzt, wobei der Isomorphismus die Knoten– und Kantenlabel beachtet. Dieses Problem ist gut studiert und in allgemeinen Graphen als NP-schwer bekannt (s. z.B. [Tar77]). Für Bäume und baumartige Graphen von beschränktem Grad ist es jedoch polynomniell lösbar [Aku93].

Da also polynomielle Algorithmen für das Subgraphen–Isomorphie–Problem unwahrscheinlich scheinen, müssen für die Anwendung Algorithmen gefunden werden, die im worst–case zwar exponentielle Zeitkomplexität haben, jedoch in der Praxis akzeptable Laufzeiten liefern.

Neben dem einfachsten Ansatz, alle potentiellen Subgraphen zu testen (brute–force), sind backtracking–Ansätze die nächstliegenden. Hier werden nicht alle Subgraphen entsprechender Größe getestet, sondern die Enumeration läuft entlang der Kantenstruktur des Graphen. Eine frühe Anwendung findet sich z.B. in [RK57]. Andere Ansätze versuchen durch iterative Partitionierung der Knotenmenge des Bindungsgraphen G_M die Anzahl der möglichen Abbildungen zu beschränken, indem z.B. Atomtypen, Knotengrad u.ä. berücksichtigt werden; in der Hoffnung, so eine eindeutige Abbildung zu finden oder aber deren Nichtvorhandensein zu garantieren ([Fig72],[Sch84]). Schließlich finden sich Algorithmen, die beides verknüpfen – also auf dem so reduzierten Graphen einen backtracking–Algorithmus starten ([Sus65],[Ull76]).

Die Nachteile dieser Ansätze bestehen vor allem darin, daß bei gegebener Substruktur eine Menge von zu testenden Molekülen nur danach partitioniert werden kann, ob ein Element der Menge diese Substruktur enthält oder nicht. Feinere Abstufungen – „enthält sie fast“ oder „enthält sie kaum“ – sind nicht möglich. Einen genaueren Überlick über Arbeiten in diesem Gebiet liefert [Bar93].

Dieser einfache Ansatz kann in der folgenden Weise erweitert werden: Anstatt nach einer gegebenen Substruktur in einem Graphen zu suchen, können Paare von Graphen G_1 und G_2 (die von Molekülen M_1 und M_2 kommen) auf ihren größten gemeinsamen induzierten Untergraphen G_{12} hin untersucht werden. Dieses Problem bezeichnen wir im Folgenden mit MCS (für maximum common subgraph). Meist betrifft die Maximalitätsbedingung die Anzahl der Kanten in G_{12}. Es ist aber auch erkannt worden, daß mit dem Übergang zum Linegraphen $L(G_i)$ (bis auf das ΔY-Problem) nach dem Satz

von Whitney [Whi32] knotenmaximale, gemeinsame induzierte Untergraphen der Linegraphen genau den kantenmaximalen induzierten Untergraphen der Ausgangsgraphen entsprechen.

In [RW02b] werden eine Reihe von exakten und approximativen Algorithmen zur Lösung des MCS beschrieben. Wir beschränken uns hier auf die wichtigsten exakten Algorithmen. Neben weiteren backtracking–Ansätzen ([McG82],[WA83]) und einer Anwendung von dynamischer Programmierung [Aku93] werden vor allem die in biologisch-chemischen Anwendung beliebten cliquenbasierten Techniken besprochen.

Das MCS Problem kann in der Folgenden Weise als Cliquensuchproblem formuliert werden: Das modulare Produkt $G_1 \diamond G_2$ zweier Graphen G_1 und G_2 ist definiert auf der Knotenmenge $V(G_1) \times V(G_2)$, wobei zwei Knoten (u_i, v_i) und (u_j, v_j) in $V(G_1 \diamond G_2)$ benachbart sind, falls

$$(u_i, u_j) \in E(G_1) \wedge (v_i, v_j) \in E(G_2)$$

oder

$$(u_i, u_j) \notin E(G_1) \wedge (v_i, v_j) \notin E(G_2).$$

Wie leicht zu erkennen ist, entsprechen nun maximale Cliquen in $G_1 \diamond G_2$ genau knotenmaximalen gemeinsamen induzierten Untergraphen in G_1 und G_2. Suchen wir die kantenmaximale Entsprechung, so ist dies, mit den schon oben geschilderten Einschränkungen, wieder mit dem Übergang zum Linegraphen möglich.

Dieser Ansatz allein führt jedoch praktisch nicht zum Ziel, da das Suchen maximaler Cliquen in den resultierenden Graphen in keiner Weise effizient gelöst werden kann – das Bestimmen der Cliquenzahl eines Graphen gehört theoretisch zu den schwierigsten kombinatorischen Optimierungsproblemen [Hås99] und auch Heuristiken zur schnellen Lösung kleiner praxisrelevanter Instanzen sind kaum bekannt.

In biologisch–chemischen Anwendungen ist es nun möglich, die resultierenden Graphen $G_1 \diamond G_2$ auszudünnen, indem nur solche Kanten betrachtet werden, die von konsistent gelabelten Knoten bzw. Kanten herkommen, d.h. wir betrachten z.B. nur Kanten, die in G_1 und G_2 zwischen Knoten verlaufen, die zu gleichen Atomtypen gehören und die vom selben Bindungstyp sind. Auf diese Weise enthält $G_1 \diamond G_2$ zwar wesentlich weniger Kanten, jedoch verlieren wir die Möglichkeit, gemeinsame induzierte Subgraphen zu finden, die auf der Ebene der ungelabelten Graphen vorhanden sind.

Das Problem des Findens der maximalen Cliquen in $G_1 \diamond G_2$ wird nun in den meisten Ansätzen (z.B. [Koc01]) mit dem Algorithmus von Bron und Kerbosch [BK73] (und leichten Abwandlungen davon) gelöst, der im Wesentlichen nicht anderes tut, als alle Cliquen zu enumerieren. Allein daran ist

sichtbar, daß ein solches Verfahren nur auf sehr dünnen Graphen Erfolg verspricht. Auch werden noch zusätzliche, speziell auf chemische Graphen zugeschnittene Heuristiken entwickelt, um $G_1 \diamond G_2$ weiter auszudünnen [RGW02].

Um nun eine Menge von Molekülen paarweise zu vergleichen, ist es sinnvoll, ein aus dem obigen Prozess abgeleitetes Ähnlichkeitsmaß einzuführen. Ähnlich wie bei den Ähnlichkeitsdefinitionen für molekulare Fingerprints (siehe 2.1.1.2) finden sich auch hier in der Literatur viele verschiedene Ansätze (siehe z.B. [RW02a]). Eine dem unten angeführten Tanimotokoeffizienten (Formel 2.1) sehr ähnliche Definition ist der sogenannte Walliskoeffizient $d(G_1, G_2)$ [WSKR01]:

$$d(G_1, G_2) = \frac{|G_{12}|}{|G_1| + |G_2| - |G_{12}|}, \text{ wobei } |G_i| := |V(G_i)| + |E(G_i)|.$$

Eine gute Übersicht über verwandte Ähnlichkeitsdefinitionen, ihre Anwendungen und ein Vergleich dieser Methoden mit den im nächsten Paragraphen geschilderten findet sich in [RW02a].

Einige Ansätze versuchen zudem, Informationen über die $3D$–Gestalt der Moleküle mitzuberücksichtigen, indem Kantenlängen spezifiziert werden, z.B. [RW03]. Information über die volle Flexibilität von Molekülen sind jedoch auf diese Weise nicht zu erfassen.

2.1.1.2 Ähnlichkeitssuche mit molekulare Deskriptoren

Molekulare Deskriptoren werden verwendet, um in abstrakter Weise Eigenschaften der betrachteten molekularen Strukturen zu beschreiben. Welche Eigenschaften in die Beschreibung eingehen, hängt sehr stark von der bearbeiteten Anwendung ab. Es finden sich in der Literatur hunderte verschiedener solcher Deskriptoren ([XB00],[Baj01]). Welche die „besten“ oder besser die „richtigen“ für welche Anwendung sind, ist Inhalt zahlreicher Arbeiten und nach heutigem Wissensstand keinesfalls eindeutig zu beantworten (siehe z.B. [WBD98]). Die kodierten Eigenschaften reichen vom einfachen Zählen von Atomen, Bindungen und Knotengraden bis hin zu komplizierten quantenmechanischen Beschreibungen, deren Berechnung sehr aufwendig ist. Es muß also abgewogen werden zwischen Einfachheit, was eine schnelle Berechnung ermöglicht, und komplizierten Modellen, deren Berechnung viel Zeit und Speicherplatz in Anspruch nimmt.

Grob können molekulare Deskriptoren nach ihrer „Dimensionalität“ unterschieden werden. $1D$-Deskriptoren beschreiben Eigenschaften, die sich auf das Molekül als Ganzes beziehen, z.B. das molekulare Gewicht. Mit $2D$-Deskriptoren werden topologische Eigenschaften gefaßt, die z.B. mit der Bindungsstruktur des Moleküls zu tun haben. $3D$-Deskriptoren schließlich bezie-

hen auch die 3–dimensionale Form der Moleküle in Betracht. Versuche, volle Information auch über die Flexibilität der betrachteten Moleküle in einem linearen Deskriptor zu speichern, wie z.B. in [MMM+99], leiden daran, daß einerseits die Deskriptoren sehr groß sind und nur aufwendig zu berechnen, und andererseits daran, daß der in dieser Weise gespeicherten Information – den $3D$–Features, also der Lage von drei oder vier speziellen Zentren des Moleküls, über alle berechneten Konformere – nicht mehr entnommen werden kann, welche davon simultan gelten, also zu einem Konformer gehören.

Andere nichtlineare Deskriptoren versuchen, die räumliche Struktur der Moleküle und relative Lage von funktionellen Gruppen in baumartigen Strukturen zu fassen, und Vergleiche von Molekülen dann durch geeignete Vergleiche dieser sogenannten feature–trees durchzuführen [RD98].

Eine spezielle Form molekularer Deskriptoren sind sogenannte Fingerprints. Diese bestehen im Allgemeinen aus binären Bitvektoren. Auf diese Weise können sowohl numerische als auch strukturelle Eigenschaften codiert werden.

Verschiedene Fingerprints unterscheiden sich stark sowohl in ihrer Länge (weniger als 100 Bits [XGB00] bis zu mehreren Millionen Bits [MMM+99]) als auch in der Art, wie die einzelnen Bits „gesetzt“ werden. In den einfachen Fällen beschreibt jedes Bit genau das Vorhandensein (1) oder Fehlen (0) einer speziellen Eigenschaft. Prominente Vertreter dieser Klasse sind MACCS–keys ([mac]). Im Gegensatz dazu steht in den Daylight Fingerprints [JW95] nicht jedes Bit für genau eine Eigenschaft, sondern für jede zu beschreibende Eigenschaft wird eine kleine Anzahl von Bits, verteilt über den gesamten Bitstring nach einem gewissen Schema, auf 1 gesetzt. Dies macht es einfacher, verschiedenste Klassen von Molekülen mit verschiedenen, für diese Klasse wichtigen Eigenschaften mit einem Bitstring gleicher Länge zu behandeln. Jedoch muß so auf die Eindeutigkeit der Aussage eines „gesetzten“ Bits verzichtet werden.

Die Idee dieser Ansätze ist, daß ähnliche Strukturen auch ähnliche Fingerprints besitzen. Die Frage, wie Ähnlichkeit (oder Verschiedenheit) von Fingerprints gemessen wird, hat ähnlich viele verschiedene Antworten gefunden wie schon das Design der Fingerprints. Das vielleicht gebräuchlichste Maß zum Vergleichen von Fingerprints ist der sogenannte Tanimotokoeffizent: Wollen wir zwei Fingerprints A und B gleicher Länge vergleichen, und sei a die Anzahl der „gesetzten“ Bits in A, b die Anzahl der „gesetzten“ Bits in B und c die Anzahl der Stellen, die bei beiden 1 sind, so schreibt sich der Tanimotokoeffizient von A und B, $TC(A, B)$, wie folgt:

$$TC(A, B) = \frac{c}{a + b - c}. \tag{2.1}$$

Dieser Wert liegt also immer zwischen 0 und 1, wobei größere Werte höhe-

re Ähnlichkeit bedeuten. Ein Überblick über andere noch gebräuchliche Ähnlichkeitsdefinitionen findet sich z.B. in [WBD98]. Dort wird auch dargelegt, daß bisher keine eindeutige Aussage getroffen werden kann, welche Kombination von Fingerprint und Ähnlichkeitsdefinitionen allgemein als überlegen angesehen werden kann. Verschiedene Anwendungen lassen allgemeine Aussagen nicht zu.

2.1.1.3 $3D$–Ähnlichkeitssuche

Im Unterschied zu den oben besprochen Ansätzen betrachten wir nun jene, die Moleküle auf ihre 3–dimensionale Ähnlichkeit hin untersuchen und die volle Flexibilität beachten. Wie auch in den vorangehend aufgeführten Paragraphen finden wir hier verschiedene Definitionen von Ähnlichkeit und viele algorithmisch verschiedene Methoden, auf Ähnlichkeit hin zu vergleichen.

Oft wird die Ähnlichkeit zweier Moleküle mit sogenannten Ähnlichkeitsindizes gemessen. Physikalische Eigenschaften der Moleküle, wie z.B. Elektronendichte oder elektrostatisches Potential, genauso wie die räumliche Überlappung der so modellierten Volumina gehen in die Berechnung der Güte einer Überlagerung ein. Prominente Vertreter solcher Indizes finden sich in [CLA80] und [HR87]. Unterschiede gibt es in den Methoden, die physikalischen Modelle numerisch zu behandeln. Wir finden z.B. gitterbasierte Verfahren [Goo85] oder analytische Methoden mit Hilfe Gauß'scher Funktionen [GHR92].

Es finden sich aber auch Ansätze, die weiterhin pharmakophor–basiert arbeiten ([MBD+93]).

Eine ganze Reihe verschiedener algorithmischer Techniken werden zur Optimierung der oben definierten Ähnlichkeiten herangezogen: Neuronale Netzwerke [FFD93] und genetische Algorithmen [JWG95], Cliquen-basierte Ansätze [MBD+93], Gradienten-Techniken [MK97] und brute-force-Ansätze, um nur einige zu nennen.

Weiterhin wird unterschieden nach starren Überlagerungen (siehe z.B. [NVK+97]), die immer nur ein Konformer pro Molekül betrachten, semi–flexiblen Verfahren (z.B. [MBD+93]), die meist in einer Vorphase Mengen von Konformeren berechnen und diese dann starr überlagern und solchen Methoden, die die Flexibilität der Moleküle in den Überlagerungsprozess miteinbinden. Vertreter der letzten Klasse sind z.B. [LL97] bzw. [LLK98] und [KHR03]. Beide Verfahren ähneln sich darin, daß sie die flexiblen Moleküle in relativ starre Teile zerlegen, diese dann aber auf verschiedenen Wegen schrittweise zu der Überlagerung hinzufügen.

Diese kurze Darstellung kann nicht vollständig sein, da die Anzahl von Publikationen auf diesem Gebiet sehr groß ist und ein erschöpfender Über-

blick den Rahmen dieser Arbeit sprengte. Eine gute Darstellung liefert der Übersichtsartikel [LL00]. Dort wird noch einmal auf die Schwierigkeit hingewiesen, die verschiedenen Verfahren miteinander zu vergleichen, einerseits wegen der z.T. sehr verschiedenen Modellierung des Problems, andererseits weil viele der bekannten Verfahren nur je einmal an sehr speziellen Datensätzen evaluiert wurden.

2.1.2 $3D$–Ähnlichkeit durch Überlagerung

Wie im vorangegangenen Paragraphen 2.1.1.3 beschrieben, finden sich in der Literatur viele verschiedene Ansätze, $3D$-Ähnlichkeit molekularer Strukturen zu formulieren. Im Folgenden wollen wir unsere Definition beschreiben und erklären.

Unser Hauptaugenmerk liegt auf der Geometrie, also der reinen 3–dimensionalen Gestalt der betrachteten Struktur. Da unsere Anwendungen (siehe Kapitel 3) nicht nur kleine Moleküle betrachten (siehe Abschnitte 3.2 und 3.3), sondern auch Teile größerer Strukturen (bestimmte Oberflächenstücke von Proteinen, siehe Abschnitt 3.1), wollen wir zuerst keinerlei Information jenseits der reinen Lage der Atome der Strukturen in unsere Ähnlichkeitsdefinition miteinbeziehen.

Wir folgen zuallererst dem Paradigma, daß eine notwendige Voraussetzung für funktionelle Ähnlichkeit eine ähnliche 3–dimensionale Gestalt ist: Sollen z.B. verschiedene Liganden an ein und derselben Stelle eines Proteins andocken – die Bindungsstelle ist also als geometrisch vorgegeben anzusehen – so müssen sie notwendig, zumindest auf dem Teil, der für diesen Bindungsprozess verantwortlich ist, sehr ähnliche räumliche Gestalt haben. Oft jedoch sind solche Bindungsstellen nicht bekannt, sondern nur ein oder wenige Kandidaten, deren funktionelle Aktivität darauf schließen läßt, daß die gewünschte Wirkungsweise vorliegt. Diese dienen sozusagen als Muster für die Suche nach weiteren potentiell ähnlich wirkenden Substanzen. Dieser allgemeine Ansatz wird auch in [LL00] beschrieben.

Ein Vorgehen nach derselben Idee wird in [PGF98] verfolgt, hier in allgemeinerer Form: Speziell ausgewählte Oberflächenstrukuren von Proteinen und ihre räumlichen Gegenstücke werden aus gemessenen Daten extrahiert mit dem Ziel, Vorhersagen für docking–Prozesse und Proteinfaltungsverhalten zu erhalten.

2.1.2.1 $3D$–Score

Bevor wir eine exakte Definition dessen angeben, was wir hier und im Folgenden als $3D$–Ähnlichkeit verstehen wollen, geben wir eine anschauliche

Beschreibung und erläutern diese inhaltlich.

Wir wollen 3–dimensionale Ähnlichkeit von molekularen Strukturen mit Hilfe einer Überlagerung im 3–dimensionalen Raum messen. Wir gehen an dieser Stelle davon aus, daß es sich bei den betrachteten Strukturen um starre Strukturen handelt. Als Eingabeinformation stehen uns nur jeweils die $3D$–Koordinaten der einzelnen Atome zur Verfügung. Vereinfacht ausgedrückt messen wir, wie gut wir die beiden Strukturen räumlich zur Deckung bringen können. Etwas formaler kann ein Überlagerungsprozess in zwei Schritte zerlegt werden. Im ersten legen wir eine Zuordung der Atome des einen Moleküls zu Atomen des anderen Moleküls fest, d.h. wir beantworten die Frage, welches Atom auf der einen Seite welchem auf der anderen (oder überhaupt keinem) zugeordnet werden soll. Im zweiten Schritt berechnen wir die starre Bewegung im Raum, die die Moleküle in der Weise zu Deckung bringt, daß die Abstände der zugeordneten Paare geeignet minimiert werden. Eine Überlagerung wird als umso besser angesehen, je mehr Atome des einen Moleküls Atomen des anderen Moleküls zugeordnet wurden und je geringer die Abstände der zugeordneten Atome nach durchgeführter räumlicher Bewegung sind. Schließlich wollen wir jeder Überlagerung einen Wert zuordnen, der ihre Qualität mißt.

Die beiden gewünschten Eigenschaften – „viele Atome“ überlagert bei „geringer“ Distanz – sind gegenläufig. Wenn wir z.B. nur ein Atom a des einen Moleküls einem Atom b des anderem Moleküls zuordnen, so können wir dies immer in der Weise tun – indem wir z.B. als Bewegung nur eine Translation des einen Moleküls derart benutzen, daß a auf b verschoben wird – daß der Abstand der zugeordneten Atome verschwindet. Verlangen wir hingegen, daß die Anzahl zugeordneter Atome steigt, so wird i.a. auch der Fehler, also die Distanz der zugeordneten Atome, steigen.

Unsere Ähnlichkeitsdefinition ist in der Lage, diese beiden gegenläufigen Eigenschaften simultan zu berücksichtigen.

Wir definieren nun formal, was wir unter einer Überlagerung verstehen wollen und wie wir diese bewerten:

Als Eingabe erhalten wir zwei Moleküle A und B mit n bzw. m Atomen, jeweils gegeben durch ihre Koordinaten im 3–dimensionalen Raum. Es ist also $A \subseteq \mathbb{R}^3$ und $B \subseteq \mathbb{R}^3$. Andere strukturelle und chemische Eigenschaften der Moleküle, wie z.B. Bindungsstruktur oder Atomtypen, werden an dieser Stelle nicht genutzt.

Eine Überlagerung setzt sich zusammen aus einer Überlagerungszuordnung und einer Überlagerungsbewegung. [1]

[1] Im Folgenden werden wir oft „Überlagerungszuordung“ und „Überlagerung“ synonym benutzen, so Mißverständnisse ausgeschlossen sind.

Definition 2.1.1 (Überlagerungszuordnug) *Eine Überlagerungszuordnung f ist eine bijektive Abbildung $f : A' \to B'$, wobei $A' \subseteq A$ und $B' \subseteq B$. Die Menge aller Überlagerungszuordnungen bezeichnen wir mit $\mathcal{F}(A, B)$. Für die Bildmenge B' schreiben wir auch $f(A')$ und für die Urbildmenge A' auch $f^{-1}(B')$. Weiterhin können wir f als eine Menge von Tupeln aus $A \times B$ schreiben mit $(a, b) \in f \Leftrightarrow f(a) = b$. Die Anzahl der jeweils zugeordneten oder überlagerten Atome nennen wir die Größe der Überlagerung, $|f| = |\{v \in A \times B : v \in f\}|$.*

Unter eine Bewegung verstehen wir das Folgende:

Definition 2.1.2 (Bewegung) *Eine Bewegung ist eine Abbildung $M : \mathbb{R}^3 \to \mathbb{R}^3$ der Form $x \mapsto Rx + t$, wobei $R \in \mathcal{O}^+(3)$ eine Rotationsmatrix [2] ist und $t \in \mathbb{R}^3$ ein Translationsvektor. Die Menge aller Bewegungen sei mit $\mathcal{B}$ bezeichnet.*

Zu gegebener Lage der Punkte von A, einer beliebiger Bewegung M_B von B und einer Überlagerungszuordnung f können wir Folgendes definieren:

Definition 2.1.3 ($rmsd(M_B, f)$) *Sei M_B eine beliebige Bewegung von B und $f \in \mathcal{F}(A, B)$. Dann bezeichnen wir mit $rmsd(M_B, f)$ die Wurzel aus dem Mittelwert der quadratischen Abstände der zugeordneten Atome:*

$$rmsd(M_B, f) := \sqrt{\frac{1}{|f|} \sum_{(u,v) \in f} |u - M_B(v)|^2}$$

Zur Notation bemerken wir, daß $|x|$ (wie in der Summe von Definition 2.1.3) die L_2–Norm von x bezeichnet. Eine Verwechslung mit der Notation $|f|$ für die Größe einer Überlagerung f sollte durch den Zusammenhang auszuschließen sein.

Unter der Überlagerungsbewegung wollen wir nun diejenige [3] Bewegung $M \in \mathcal{B}$ verstehen, die $rmsd(M_B, f)$ zu gegebenem f minimiert:

Definition 2.1.4 (Überlagerungsbewegung M_f) *Die Überlagerungsbewegung M_f zur Überlagerungszuordnung f ist diejenige Bewegung $M \in \mathcal{B}$, die $rmsd(M_B, f)$ minimiert, d.h.*

$$rmsd(M_f, f) \leq rmsd(M_B, f) \; \forall \; M_B \in \mathcal{B}.$$

[2] $R \in \mathcal{O}^+(3) \Leftrightarrow \mathrm{R} \in \mathbb{R}^{3\times 3}$ ist orthonormal mit $det\, R = 1$

[3] Diese Bewegung ist nicht notwendig eindeutig, jedoch der minimale Wert für $rmsd(M_B, f)$ ist eindeutig. Wir wählen also stets eine Bewegung, die diesen Wert minimiert.

Mit $rmsd^*(f)$ bezeichnen wir dann den kleinstmöglichen Wert, den rmsd annehmen kann:

$$rmsd^*(f) := rmsd(M_f, f) := \min_{M_B \in \mathcal{B}} rmsd(M_B, f).$$

Aus anderen Anwendungszusammenhängen ist bekannt, siehe [Ume91], daß M_f existiert und wie es effizient, also in polynomieller Laufzeit in der Anzahl der Atome, berechnet werden kann.

Schließlich können wir unsere Ähnlichkeitsdefinition angeben:

Definition 2.1.5 ($score(A, B)$) *Sei nun $f \in \mathcal{F}(A, B)$ und M_f gegeben, so definieren wir den score von f vermöge*

$$score(f) := \frac{|f|}{\min(n, m)} \exp(-rmsd^*(f))$$

Schließlich sei

$$score(A, B) := \max_{f \in \mathcal{F}(A,B)} score(f).$$

Wir können nun sehen, wie der einem Paar von Punktmengen A und B zugeordnete Ähnlichkeitswert $score(A, B)$ die oben angesprochenen gegenläufigen Eigenschaften integriert: Der erste Term des Produktes, $\frac{|f|}{\min(n,m)}$, wächst mit der Größe der zugeordneten Punktmengen. Der zweite Term $\exp(-rmsd^*(f))$, der den Fehler mißt, ist umso größer, je besser die zugeordneten Punktepaare durch die Bewegung M_f übereinander gelegt werden können.

Wie leicht zu erkennen ist, gilt immer $0 \leq score(A, B) \leq 1$. Die Formulierung des ersten Terms erlaubt es auch, Teilstrukturen einer größeren Struktur zu finden – eine Eigenschaft, die vielen anderen Ähnlichkeitsdefinitionen fehlt.

Die in Definition 2.1.5 angegebene Form der Ähnlichkeitsdefinition kann in gewisser Weise als allgemeine Form einer Klasse von ähnlichen Maßen gesehen werden:

- Im ersten Term kann das Verhältnis z.B. auch im Vergleich zur größeren Struktur genommen werden, was eine Tendenz hin zu gleichgroßen Molekülen unterstützt.

- Die Klasse der Überlagerungszuordnungen kann eingeschränkt werden, sei es durch strukturelle Zusatzinformationen, sei es durch chemische oder physikalische Eigenschaften der Atome.

- Der Einfluß des Fehlers im zweiten Term kann durch die Wahl einer anderen Basis verändert werden.

Eine Zielfunktion dieser Art findet sich implizit schon in [PGF98]. Die dort angegebene Heuristik, den *score* zu gegebenem Paar von Strukturen zu approximieren, zeigt jedoch Schwächen (siehe auch Abschnitt 3.1).

Ziel des Folgenden ist es nun, verschiedene exakte und heuristische Verfahren zur Berechnung von $score(A, B)$ darzustellen. In Kapitel 3 zeigen wir praktische Anwendungen der hier beschriebenen Algorithmen in verschiedenen Anwendungszusammenhängen.

2.1.2.2 Ein exakter Algorithmus

Ein erstes Ziel ist es, einen exakten Algorithmus anzugeben, der $score(A, B)$ wie in Definition 2.1.5 berechnet. Obwohl die Komplexität dieses Problems noch nicht endgültig geklärt ist, gehen wir davon aus, daß es nicht in P liegt, wir also nicht erwarten können, einen exakten Algorithmus mit polynomieller Laufzeit zu finden.

Ein naheliegender erster Ansatz ist ein Branch and Bound–Verfahren.

In der allereinfachsten Form – also ohne jede spezielle Strategie für die Verzweigungsreihenfolge und ohne Berechnung von Schranken zum Abschneiden des Enumerationsbaumes – bedeutet dies, alle möglichen Überlagerungszuordnungen von Atomen in A auf Atome in B zu enumerieren. Da jedoch die Anzahl dieser Zuordnungen sehr schnell steigt [4], können nur sehr kleine Instanzen (mit weniger als 10 Atomen) auf diese Weise noch in akzeptabler Zeit berechnet werden.

Eine genauere Betrachtung der Zielfunktion in Definition 2.1.5 zeigt jedoch, daß effiziente Schranken zur Beschneidung des Enumerationsbaumes nicht leicht zu erhalten sind. Dies liegt im Wesentlichen daran, daß die Zielfunktion weder monoton noch additiv ist. Letzteres sagt aus, daß sich die Güte einer Lösung, die sich aus zwei Teillösungen ergibt, nicht aus der Güte der Teillösungen berechnen läßt. Anschaulich bedeutet dies, daß zwar lokal gute Teilüberlagerungen existieren können, die jedoch global nicht simultan bestehen. Fehlende Monotonie bedeutet, daß auch aus nur mäßigen Teillösungen noch optimale Gesamtlösungen werden können (da z.B. für Teillösungen, die erst einen kleinen Teil der Atome betrachtet haben, der erste Term der Zielfunktion den Gesamtwert nach oben beschränkt, auch wenn die Teilüberlagerung optimal ist, also $rmsd = 0$ gilt).

[4] Wenn wir o.B.d.A. annehmen, daß $n \leq m$, so erhalten wir $\sum_{k=1}^{n} \binom{n}{k}\binom{m}{k}k!$ viele Möglichkeiten, was z.B. für $n = m = 10$ schon $2.3 \cdot 10^8$ und für $n = m = 15$ schon $3 \cdot 10^{14}$ ergibt.

Algorithmus 1 : Initialisierung

Input : - Matrizen zu A (n Punkte) und B (m Punkte), $n \leq m$
- Vektor f für die Zuordnung
- Vektor N für Verfügbarkeit von Punkten in B
- $f[0]$:= Größe der Zuordnung
- $f[i] = j \Leftrightarrow i$-ter Punkt aus A ist j-tem Punkt aus B zugeordnet
- $N[k] = k \Leftrightarrow k$-ter Punkt in B noch nicht zugeordnet

```
/*Initialisierung                                          */
/*leere Zuordnung, jeder Punkt in B verfügbar              */
```

1 **for** $i \leftarrow 0$ **to** n **do**
  $f[i] = 0$
2 **for** $i \leftarrow 0$ **to** m **do**
  $N[i] = i$
3 *bester_wert* $\leftarrow -1$

```
/*sortiere Punkte von A nach Distanzkriterium              */
/*siehe Verzweigungsstrategien                             */
```

4 *sort_dist(A)*

Schranken Eine sehr einfache, jedoch vergleichsweise sehr effektive Strategie, die Güte der aus einer Teillösung noch zu erweiternden Gesamtlösung abzuschätzen, kann folgendermaßen aussehen: Wir nehmen an, daß alle noch nicht zugeordneten Punkte in A in der Weise zugeordnet werden können, daß kein zusätzlicher Fehler entsteht, d.h. die partielle Lösung f habe $rmsd^*(f) = \sqrt{\frac{x}{|f|}}$, also $score(f) = \frac{|f|}{n} \exp(-\sqrt{\frac{x}{|f|}})$, so gilt für jede Erweiterung f' von f, daß $score(f') \leq \frac{|f'|}{n} \exp(-\sqrt{\frac{x}{|f'|}})$. (O.B.d.A gelte auch hier $n \leq m$.)

Um eine zweite Strategie darzustellen, muß zuerst kurz die Vorgehensweise des Algorithmus in [Ume91] betrachtet werden, der die optimale Bewegung M_f zu gegebenem f berechnet. Diese besteht aus zwei Teilen: Einer Translation t und einer Rotation R. Die Translation t berechnet sich dadurch, daß wir die jeweiligen Schwerpunkte der zugeordneten Punkte von $A' \subset A$ bzw. $B' \subset B$ aufeinander verschieben. Wir sagen, daß A festbleibe und B verschoben werde. Danach wird eine Punktmenge, wieder B, gemäß R um den gemeinsamen Schwerpunkt gedreht, wobei sich R berechnet mit Hilfe einer Singulärwertzerlegung eines Produktes von Matrizen mit den Koordinaten aus A' resp. B'. Insbesondere sind also t und R separat zu berechnen. Dies machen wir uns in der folgenden Schranke zunutze:

Algorithmus 2 : Score-Berechnung mit Schranke 1

Input : - Punktmengen A und B, Überlagerungszuordnung f

`/*berechne optimale Bewegung` M `*/`

$M \leftarrow berechne\_M(f, A, B)$
$sum \leftarrow berechne\_summe\_distanzquadrate(M, A, B, f)$
$anz \leftarrow 0$
for $j \leftarrow 1$ **to** $|f|$ **do**
  if $f[j] > -1$ **then**
    $anz \leftarrow anz + 1$
$val_2 \leftarrow \sqrt{\frac{sum}{anz+n-|f|}}$
$val_1 \leftarrow \frac{anz+n-|f|}{n}$
$val \leftarrow val_1 \cdot e^{-val_2}$
return val

Erweitern wir eine Überlagerung $f : A \to B$ mit $rmsd^*(f) = \sqrt{\frac{x}{|f|}}$ zu $f' : A^+ \to B^+$, wobei $A \subset A^+$ und $B \subset B^+$ gelte, so können wir $rmsd^*(f')$ schreiben als

$$rmsd^*(f') = \sqrt{\frac{\sum_{(a_i,b_i)\in f} |a_i - b_i|^2 + \sum_{(p_j,q_j)\in f'\setminus f} |p_j - q_j|^2}{|f'|}}. \tag{2.2}$$

Die erste Summe unter der Wurzel mißt den Fehleranteil der Paare von Punkten, die schon durch f zugeordnet waren. Dieser beträgt mindestens x, wird sich aber im Allgemeinen vergrößern auf einen Wert $x' \geq x$. Diesen Sachverhalt untersuchen wir in Lemma 2.1.1.

Ziel der folgenden Betrachtungen ist es, den Fehleranteil der in $f' \setminus f$ neu zugeordneten Paare von Punkten abzuschätzen. Wir skizzieren zuerst die Hauptidee:

Wir betrachten den einfachen Fall, daß f nur um ein Paar (p, q) erweitert wird. Die optimale Bewegung $M_{f'}$ zu f' verschiebt den Schwerpunkt S_{B^+} von B^+ vermöge der Translation t' auf den Schwerpunkt S_{A^+} von A^+, bevor die Rotation R' die Punktmenge B^+ um den nun gemeinsamen Schwerpunkt von A^+ und B^+ dreht.

Weiterhin gehen wir davon aus, daß bevor $M_{f'}$ ausgeführt wurde, die Lage der Punktmengen A^+ und B^+ der entsprach, die sie nach der Ausführung von M_f eingenommen hat. Bezüglich dieser Lage bezeichnen wir den Abstand von p und q mit d, $d = |p - q|$. Bezeichnen wir die Differenz der Abstände von S_{A^+} und S_{B^+} zu S, gemessen vor der Ausführung von $M_{f'}$, mit v', also $v' = ||S - S_{A^+}| - |S - S_{B^+}||$, so gilt sicherlich $|t'| \geq v'$.

Es gilt nun, daß der Abstand von p und $M_{f'}(q)$ nicht kleiner sein kann als $|d - v'|$, da die Translation t' die Punkte p und q höchstens um den Betrag näher gebracht haben kann, um den sich die Schwerpunkte S_{A^+} und S_{B^+} im Vergleich zu S verschoben haben. Somit gilt: $|p - M_{f'}(q)| \geq |d - v'|$.

Mit dieser Argumentation erhalten wir eine Möglichkeit, den Fehleranteil der Paare von Punkten aus $f' \setminus f$ abzuschätzen, ohne die Zuordnung von f' zu kennen. Wir gehen wie folgt vor:

Sei $A^+ \setminus A = \{p_1, p_2, \ldots, p_k\}$ und $B^+ \setminus B = \{q_1, q_2, \ldots, q_j\}$. O.B.d.A. sei $k \leq j$. Wir wollen eine untere Schranke für den Fehleranteil aller Paare der Gestalt (p_i, q_j) für jede mögliche echte Erweiterung f' von f angeben.

Für jede mögliche Anzahl von zu f hinzukommenden Paaren, also für alle $1 \leq l \leq k$, berechnen wir den mindestens auftretenden Fehleranteil auf die folgende Weise:

Wir berechnen die Abstände der p_i, $1 \leq i \leq k$, und der q_r, $1 \leq r \leq j$, zu S und bestimmen dann, aufsteigend sortiert, die l kleinsten Abstände $d_t = |p_t - q_t|$. Hierbei kann es vorkommen, daß einzelne p_i oder q_j mehrfach benutzt wurden, was zu einer nicht zulässigen Überlagerung führte. Da wir jedoch hier nur eine untere Schranke berechnen, und auch nicht alle möglichen zulässigen Zuordnungen enumerieren können, stört uns dies nicht.

Wie in der obigen Darstellung der Erweiterung um ein Paar können wir wie folgt argumentieren: Die Translation t' zur Überlagerung f' – die wir nicht kennen – kann im besten Fall, wollen wir einen sehr kleinen Fehler erhalten, alle Paare (p_t, q_t) von Punkten simultan näher zueinander verschieben, höchstens allerdings um den Betrag v'. (d.h. $|p_t - t'(q_t)| < |p_t - q_t| \quad \forall t$. Die Rotation R' ändert nichts mehr an den Abständen) Da wir f' nicht explizit kennen, so auch nicht v', können wir nur sagen, daß der Anteil des Fehlers in Gleichung 2.2, der durch die Paare in $f' \setminus f$ hinzukommt, mindestens

$$v = \sum_{t=1}^{l} |\max(d_t - v', 0)|^2. \tag{2.3}$$

beträgt. Wir müssen hier beachten, daß es für eine Überlagerung sinnvoll sein kann, einzelne Paare, die sehr kleine Abstände haben, nicht zuzuordnen. So ist es zu erklären, daß wir für $v' > d_t$ keinen Fehler mehr addieren dürfen.

Um schließlich die gewünschte Schranke zu erhalten, führen wir also obige Prozedur in der Weise durch, daß wir dasjenige v' numerisch bestimmen, das einen Fehleranteil v liefert, der addiert mit dem Anteil x' , der zu den Paaren von f gehört (siehe Lemma 2.1.1), minimal ist (siehe auch Algorithmus 3).

Lemma 2.1.1 (Verschiebungsverlust) *Seien $A = \{a_1, \ldots, a_n\}$ und $B = \{b_1, \ldots, b_n\}$, $n \in \mathbb{N}$, zwei Mengen von Punkten in $\mathbb{R}^3$ und $f : A \to B$ eine Zuordnung mit $f(a_i) = b_i \forall i \in \{1, \ldots, n\}$. Sei M_f die Überlagerungsbewegung zu f und $rmsd^*(f) = \sqrt{\frac{x}{n}}$ mit $\sum_{i=1}^{n} |a_i - b_i|^2 = x$, d.h. die Koordinaten von A und B entsprechen denen nach der optimalen Bewegung M_f. Der gemeinsame Schwerpunkt der Mengen A und B sei S.*

Weiterhin sei $M_{f'} : x \mapsto R'x + t'$ die Überlagerungsbewegung zu einer Erweiterung $f' : A^+ \to B^+$, $A \subset A^+, B \subset B^+$ von f. Seien S_{A^+} und S_{B^+} die Schwerpunkte von A^+ und B^+ und sei $u = ||S - S_{A^+}| - |S - S_{B^+}||$.

Dann gilt:

$$x' := \sum_{i=1}^{n} |a_i - b'_i|^2 \geq x + n \cdot u^2, \tag{2.4}$$

wobei $B' := \{b'_1, \ldots, b'_n\} := M_{f'}(B)$.

Beweis: Wir zeigen zuerst, daß sich nach vollzogener Bewegung $M_{f'}$ die Schwerpunkte S_A und S_B der Mengen A und B, die sich zu Beginn beide in S befinden, mindestens den Abstand u haben. Ist dies gezeigt, läßt sich der zusätzliche Fehler leicht abschätzen.

$M_{f'}$ verschiebt S_{B^+} auf S_{A^+} und dreht dann B^+ gemäß R'. Wir zerlegen diese Bewegung wie folgt: Verschiebe S_{B^+} nach $S = S_A$, drehe gemäß R' und verschiebe dann S_{B^+} nach S_{A^+}. Sei also $t_1 = \overrightarrow{S_{B^+}S_A}$, $t_2 = \overrightarrow{S_A S_{A^+}}$, so können wir t' schreiben als $t' = t_1 + t_2$. Nach Annahme ist $||t_1| - |t_2|| = u$. Das Bild $M_{f'}(S_B)$ entsteht, indem S_B also erst vermöge t_1 verschoben wird, dann gedreht, was den Abstand zu S_A nicht ändert, und schließlich um t_2 verschoben wird. Somit gilt, daß $|S_A, M_{f'}(S_B)| \geq u$.

Für den Rest des Beweises können wir uns nun wieder den Mengen A und B zuwenden. Wir wissen nun, daß $M_{f'}(S_B)$ einen Abstand von S_A hat, der mindestens u beträgt. Wir können uns also vorstellen, daß $B' = f'(B)$ aus B entsteht durch eine Drehung R'' um $S = S_A$ und eine Verschiebung t'' mit $|t''| \geq u$.

Wir wissen, daß mit $R''(B) =: B_{R''} = \{b_i^{R''}, \ldots, b_n^{R''}\}$, gilt:

$$\sum_{i=1}^{n} |a_i - b_i|^2 \leq \sum_{i=1}^{n} |a_i - b_i^{R''}|^2, \tag{2.5}$$

da A und B sich bezüglich der durchschnittlichen Summe der quadratischen Abstände in optimaler Lage befinden.

Weiterhin gilt, daß A und $B_{R''}$ einen gemeinsamen Schwerpunkt haben. Betrachten wir die a_i, $b_i^{R''}$ und t'' komponentenweise (und schreiben $a_i = (x(a_i), y(a_i), z(a_i))$, analog für $b_i^{R''}$ und t'') und denken uns (zur technischen

Vereinfachung) $S = (0, 0, 0)$, so gilt:

$$\sum_{i=1}^{n} x(a_i) = 0 \text{ und} \sum_{i=1}^{n} x(b_i^{R''}) = 0 \tag{2.6}$$

(analog für die y– und z–Komponenten). Somit ist wegen $M_{f'} : x \mapsto R''x + t''$

$$\begin{aligned}\sum_{i=1}^{n}(x(a_i) - x(b_i'))^2 &= \sum_{i=1}^{n}(x(a_i) - x(b_i^{R''}) - x(t''))^2 \\ &= \sum_{i=1}^{n}((x(a_i) - x(b_i^{R''}))^2 - \sum_{i=1}^{n} 2(((x(a_i) - x(b_i^{R''}))x(t') + n \cdot x(t'')^2 \\ &\overset{2.6}{=} \sum_{i=1}^{n}(x(a_i) - x(b_i^{R''}))^2 + n \cdot x(t'')^2\end{aligned} \tag{2.7}$$

Nun gilt wegen 2.5, 2.7 und da $|t''| \geq u$:

$$\begin{aligned}\sum_{i=1}^{n} |a_i - b_i'|^2 &= \sum_{i=1}^{n} |a_i - b_i^{R''}|^2 + n \cdot |t''|^2 \\ &\geq \sum_{i=1}^{n} |a_i - b_i|^2 + n \cdot u^2\end{aligned}$$

□

Verzweigungsstrategien Die Reihenfolge, in der wir die Zuordnungen enumerieren, war bisher noch nicht festgelegt. Zwei Strategien haben sich hier als hilfreich erwiesen.

Um früh in der Enumeration tatsächlich schon Schranken zu erhalten, beginnen wir die Zuordnung für jeden Punkt in A damit, daß wir ihn als nicht zugeordnet betrachten, bevor wir ihm einen Punkt in B zuweisen. Dieses Vorgehen findet sich in Algorithmus 4 in den Zeilen 4–5. Dies hat zur Folge, daß die so aufgezählten Zuordungen am Anfang zwar von der Größe her klein sind, ihnen jedoch konkret ein *score*-Wert zugeordnet werden kann, der in vielen Fällen besser ist als eine Schranke der oben genannten Art, in der noch fast alle Punkte unbestimmt, also potentiell noch sehr gut zuzuordnen sind.

Als zweites haben wir die Punkte von A in geeigneter Weise sortiert, und dann in diese Reihenfolge enumeriert. Die Sortierung beabsichtigt, Punkte in der Weise anzuordnen, daß zuerst jene betrachtet werden, die paarweise „weit“ voneinander entfernt liegen. Falls eine gute Überlagerung existiert, die

Algorithmus 3 : Score-Berechnung mit Schranke 2

Input : - Punktmengen A und B, Überlagerungszuordnung f

1 $max\_val \leftarrow -1$
2 $M = berechne\_M(f, A, B)$
3 $sum = berechne\_summe\_distanzquadrate(M, A, B, f)$
`/*für jede mögliche Anzahl */`
4 **for** $i \leftarrow 0$ **to** $n - |f|$ **do**
5 $u = berechne\_zusatzfehler\_in\_A(i, A, B, f)$
6 $v = berechne\_zusatzfehler\_durch\_d_i(i, A, B, f)$
7 $anz \leftarrow 0$
8 **for** $j \leftarrow 1$ **to** $|f|$ **do**
if $f[j] > -1$ **then**
$anz \leftarrow anz + 1$
9 $val_2 = \sqrt{\frac{sum+u+v}{anz+i}}$
10 $val_1 = \frac{anz+i}{n}$
if $val \leftarrow val_1 \cdot e^{-val_2} > max\_val$ **then**
$max\_val \leftarrow val$
return max_val

diese weit auseinanderliegenden Punkte zuordnet, so ist diese räumlich stark festgelegt. Geschieht dies am Anfang des Enumerationsprozesses – so die Idee – werden andere, nicht zur optimalen Lösung führende Zuordnungen nicht mehr so leicht akzeptiert, da sie dann große Fehler vorweisen.

Mit diesen eingebauten Schranken und Verzweigungsstrategien verläuft der exakte Algorithmus nach Initialisierung (siehe Algorithmus 1) wie in Algorithmus 4.

Im Vergleich zur puren Enumeration können mit diesen Schranken und Verzweigungsstrategien große Fortschritte erzielt werden. War es zuvor nicht möglich, Instanzen mit nur 10 Atomen in akzeptabler Zeit zu lösen (Laufzeit im Bereich von wenigen Minuten auf derzeit aktuellen PC's), so können wir nun molekulare Strukturen mit bis zu 16 Atomen in ca. zwei Minuten exakt vergleichen.

2.1.3 $3D$-Überlagerungsalgorithmus I

Der in Abschnitt 2.1.2.2 beschriebene Algorithmus löst das Überlagerungsproblem zwar exakt, ist aber für eine praktische Anwendung noch bei weitem zu langsam. Auch sind die behandelbaren Molekülgrößen nicht ausreichend.

In diesem Kapitel beschreiben wir ein Verfahren, welches diese beiden Mängel behebt: Wir können größere Instanzen wesentlich schneller lösen.

Algorithmus 4 : Branch and Bound–Algorithmus

```
   Input : - Punktmengen A und B
   - Überlagerungszuordnung f
 1 if f[0] = n then
       /*Alle Punkte aus A zugeordnet                                  */
 2     wert ← SCORE(f, A, B)
 3     if wert > bester_wert then
           bester_wert ← wert
   else
       /*Fixiere Punkt in A als partnerlos                             */
 4     f[0] ← f[0] + 1
 5     f[f[0]] ← −1
 6     wert ← SCORE_Schranke_i(f, A, B)
 7     if wert > bester_wert then
           berechne_score(f) /*rekursiv weiter                         */
       /*Lösche Zuordnung                                              */
 8     f[f[0]] ← 0
 9     f[0] ← f[0] − 1
10      for i ← 1 to m do
           if N[i] = i then
               /*Punkt i in B verfügbar                                */
11             f[0] ← f[0] + 1
12             f[f[0]] ← i
13             N[i] ← −1
14             wert ← SCORE_Schranke_i(f, A, B)
               if wert > bester_wert then
                   berechne_score(f)
               /*Lösche Zuordnungen                                    */
15             f[f[0]] ← 0
16             f[0] ← f[0] − 1
17             N[i] ← i
```

Der Preis dieser Verbesserung ist, daß wir auf eine Gütegarantie verzichten müssen: Wir erhalten keinen beweisbar exakten Algorithmus mehr, sondern nur eine Heuristik, also ein Verfahren, das beobachtbar meist sehr gute Resultate liefert, die sehr nahe an der optimalen Lösung liegen, jedoch kann es in seltenen Fällen passieren, daß das nun erhaltene Ergebnis vom Optimum abweicht. Wir können weiterhin nicht garantieren, wie groß diese Abweichung im schlechtesten Fall sein wird.

Mittlerweile wurde in [Kir03] gezeigt, daß das Überlagerungsproblem ein

PTAS, also ein Polynomzeit–Approximationsschema, besitzt, d.h. das Überlagerungsproblem in der hier betrachteten Form kann beliebig gut approximiert werden, wobei die Laufzeit dazu nur polynomiell von der Eingabe – also von der Anzahl der Atome – und der zu erreichenden Approximationsgüte abhängt.

Bevor wir unseren Algorithmus im Detail beschreiben, geben wir einen groben Überblick. Hauptproblem beim exakten Algorithmus in 2.1.2.2 ist die Laufzeit, die exponentiell mit der Anzahl der Atome steigt, jedoch können wir kleine Instanzen mit wenigen Atomen noch sehr schnell lösen.

Im hier beschriebenen Verfahren wird sich dieses Verhalten zunutzte gemacht. Vereinfacht ausgedrückt versuchen wir, die größeren nun zu behandelnden Instanzen in kleinere artifizielle Instanzen zu transformieren. Diese können wir mit dem exakten Verfahren schnell lösen. Schließlich heben wir die so erhaltene Lösung wieder hoch auf die Originalinstanz.

Soll ein derartiger Ansatz gute Resultate liefern, so müssen die angegebenen drei Phasen einige Bedingungen erfüllen. Zum ersten muß die Transformation im ersten Schritt so beschaffen sein, daß die relevante räumliche Struktur der Originalinstanzen auch noch durch die sehr viel kleineren artifiziellen Instanzen transportiert wird – eine Vergröberung ohne großen Informationsverlust ist also das Ziel.

Gelingt dies, so ist der Erfolg des zweiten Schrittes gesichert. Die kleinen künstlichen Instanzen können schnell und exakt gelöst werden, wie wir in 2.1.2.2 gesehen haben.

Für das Gelingen der dritte Phase – das Hochheben der so erhaltenen Lösung auf die Originalinstanzen – müssen wir davon ausgehen können, daß nur noch Feinkorrekturen notwendig sind, d.h. die Lage der Moleküle, dargestellt durch die vereinfachten, artifiziellen Pseudomoleküle nach dem zweiten Schritt, entspricht im Wesentlichen schon genau jener, die wir für die Ausgangsinstanzen suchen. Ein Rückgängigmachen des Vergröberungsprozesses liefert eine räumliche Lage der Moleküle zueinander in der Art, daß jeweils solche Atome, die in einer optimalen – oder zumindest sehr guten – Überlagerung einander zugeordnet werden, nun sehr nahe beieinander liegen. In diesem Fall kann nun diese Zuordnung lokal, d.h. schnell und ohne aufwendige Enumeration, durchgeführt werden.

Im Folgenden beschreiben wir die einzelnen Phasen des Algorithmus im Detail (2.1.3.1–2.1.3.3). Die Wahl verwendeter Parameter für verschieden Szenarien wird skizziert (2.1.3.4) wie auch eine öffentlich zugängliche Implementation des Verfahrens (2.1.3.5).

Zur Veranschaulichung betrachten wir Abbildungen eines exemplarischen Laufes des Algorithmus bei Vergleich zweier Moleküle (siehe Tabelle 2.1).

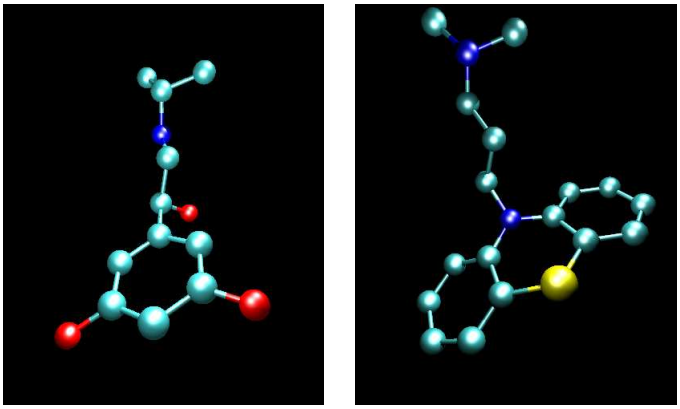


Tabelle 2.1: Zwei Moleküle im Vergleich: Orciprenalinsulfat (links) und Promazinehydrochlorid.

2.1.3.1 Phase 1: Vergröberung

Wie bereits oben angedeutet ist das Ziel dieser Phase des Algorithmus, die Moleküle in einer Weise zu vergröbern, daß die neu entstehenden artifiziellen Pseudomoleküle die räumliche Struktur so gut als möglich erhalten, jedoch wesentlich kleiner sind in dem Sinne, daß die Anzahl ihrer Pseudoatome deutlich kleiner ist als die Anzahl der Atome der Ausgangsmoleküle.

Hierzu benutzen wir ein einfaches iteratives Verfahren, welches hierarchischen Clusterungsverfahren ähnlich ist. Wir starten mit der Ausgangsinstanz A, einer Menge von n Punkten in $\mathbb{R}^3$, und nennen jeden Punkt ein Pseudoatom (mit Gewicht 1). In jedem Schritt der Iteration werden die beiden Pseudoatome zu einem neuen Pseudoatom verschmolzen, die sich räumlich am nächsten liegen. Der so neu entstandene Punkt erhält als Koordinaten den gewichteten Schwerpunkt der Pseudoatome aus denen er hervorgeht und als Gewicht die Summe ihrer Gewichte. Die beiden Pseudoatome, die den neuen Punkt bilden, werden hernach gelöscht.

Dieser Schritt, der also die Anzahl der Pseudoatome immer um 1 verringert, wird so lange durchgeführt, bis eine vorher definierte Anzahl $r < n$ erreicht ist.

In Tabelle 2.2 sehen wir einen exemplarischen Ablauf dieser Phase mit $r = 3$.

2.1.3.2 Phase 2: Exakter Algorithmus auf artifiziellen Instanzen

Die in Phase 1 entstandenen Pseudoatome werden nun mit dem exakten Algorithmus 2.1.2.2 überlagert, wobei die Gewichte der Pseudoatome in natürlicher Weise berücksichtigt werden: Wird ein Pseudoatom vom Gewicht

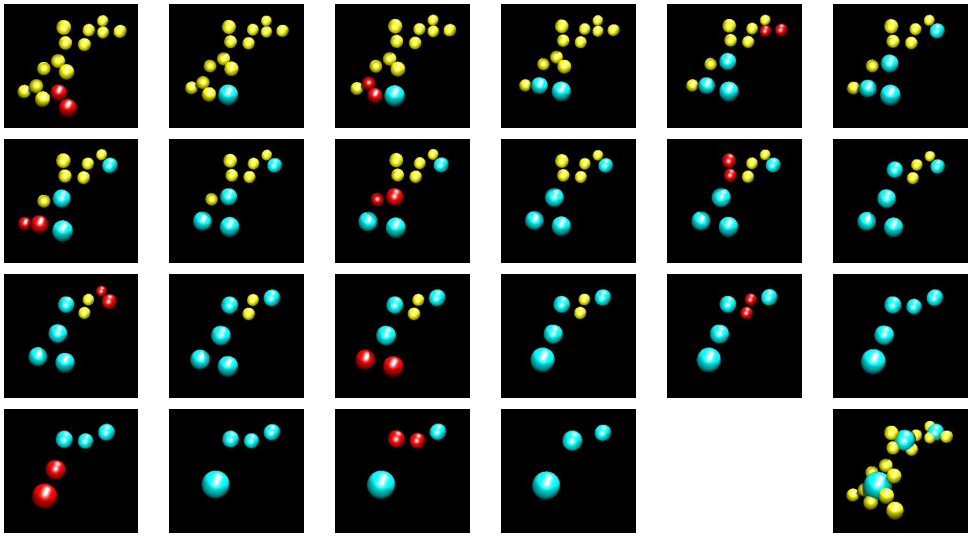


Tabelle 2.2: Exemplarischer Ablauf von Phase 1 (von links oben nach rechts unten): Wir starten mit den (gelben) Ausgangsatomen. Die roten Punkte sind die Pseudoatome, die im nächsten Schritt zum neuen Pseudoatom (cyan) verschmolzen werden. Die Größe der Punkte symbolisiert ihr Gewicht. Das letzte Bild zeigt die resultierenden drei Pseudoatome zusammen mit der ursprünglichen Instanz.

l_1 einem anderen vom Gewicht l_2 zugeordnet, wobei $l_1 \leq l_2$ gelte, so trägt dieses Paar zum ersten Term der Score–Berechnung (Definition 2.1.5) mit dem Beitrag $\min(l_1, l_2)$ im Zähler bei, wobei sich der Nenner bildet als die Summe der Gewichte der Pseudoatome im kleineren Molekül.

Für unser Anwendungsbeispiel ist diese Phase in Tabelle 2.3 gezeigt.

2.1.3.3 Phase 3: Hochheben der Lösung auf die Originalinstanzen

In der dritten Phase des Algorithmus geht es nun darum, die optimale Lösung auf den kleineren künstlichen Instanzen zu einer Lösung auf den Originalinstanzen zu machen. Der generellen Idee des Ansatzes folgend, daß die Lage der Ausgangsmoleküle, induziert durch die Lösung in Phase 2, schon fast die richtige ist, gehen wir wie folgt vor:

Wiederum in mehreren Iterationen fixieren wir Zuordnungen der Ausgangsatome zueinander.

- **Schritt** 0 Zu Beginn seien alle Atome nicht fixiert.
- **Schritt** 1 Sortiere die noch nicht zugeordneten, d.h. die noch nicht fixierten Atome des einen Moleküls aufsteigend nach ihren Abständen

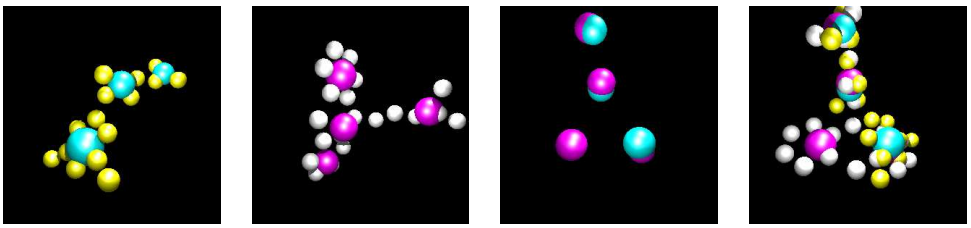


Tabelle 2.3: Exemplarischer Ablauf von Phase 2: Die beiden Abbildungen links zeigen die ursprünglichen Instanzen (in gelb bzw. weiß) und die aus ihnen jeweils in Phase 1 erhaltenen artifiziellen Instanzen (in cyan (mit $r = 3$) und magenta (mit $r = 4$)). Rechts sehen wir die optimale Überlagerung der artifiziellen Instanzen und die induzierte Lage der Originalinstanzen.

zu ihren nächsten noch nicht fixierten Nachbarn des anderen Moleküls. Wähle die ersten s Atome in dieser Reihenfolge, wobei $s < n$ eine kleine Konstante sei.

- **Schritt** 2 Enumeriere alle möglichen Zuordnungen dieser s Atome des einen Moleküls zu der Menge ihrer nächsten Nachbarn im anderen Molekül. (Auch zulässig sind Zuordnungen, in denen einzelne Atome keinen Partner zugewiesen bekommen.) Führe die durch diese Zuordnung – alle bereits fixierten Atome vereint mit den s ausgewählten dieser Runde der Iteration – induzierte optimale Bewegung durch und wähle schließlich die Zuordnung, die den besten Score–Wert liefert (auf dieser eingeschränkten Instanz der bis zu diesem Zeitpunkt fixierten Atome vereint mit den s neu zugeordneten Atomen).

- **Schritt** 3 Fixiere die in Schritt 2 gewählte Zuordnung und führe die durch die fixierten Atome induzierte optimale Bewegung für das gesamte Molekül durch. Falls noch nicht alle Atome fixiert sind, gehe zu Schritt 1.

Ein kleiner Ausschnitt aus Phase 3 soll in Tabelle 2.4 veranschaulicht werden.

In Tabelle 2.5 schließlich sehen wir das Ergebnis nach Beendigung des Algorithmus auf unserer Beispielinstanz.

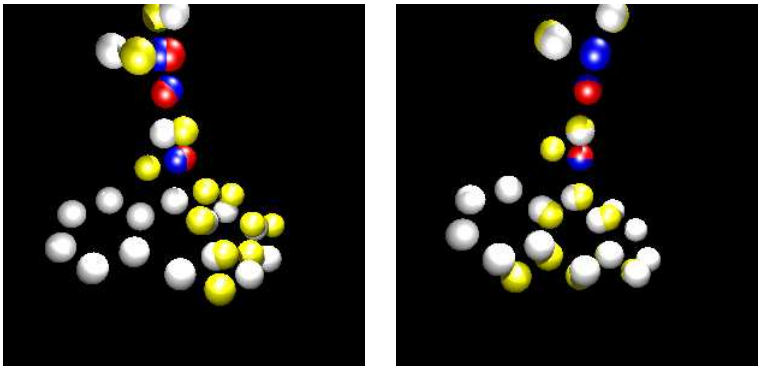


Tabelle 2.4: Exemplarische Darstellung der ersten Iteration von Phase 3: Im linken Bild sehen wir die Auswahl der $s = 3$ Atome des einen Moleküls (rot), die die kleinsten Abstände zu ihren nächsten Nachbarn im anderen Molekül haben (blau). Die Lage der Moleküle zueinander ist die nach Beendigung von Phase 2. Alle noch nicht fixierten und in dieser Phase nicht ausgewählten Atome sind weiß bzw. gelb. Im rechten Bild ist die Lage der gesamten Moleküle zueinander gezeigt, nachdem die durch die ausgewählten und fixierten Atome (rot und blau) induzierte Bewegung auf die Originalinstanzen angewendet wurde. In diesem Beispiel kann gut beobachtet werden, daß schon nach dieser einen Runde der Iteration die Lage der Ausgangsmoleküle zueinander fast perfekt ist.

2.1.3.4 Parameterwahl

Wie schon in der Beschreibung der drei Phasen des Algorithmus (2.1.3.1–2.1.3.3) zu erkennen ist, spielt die Wahl der Parameter r in Phase 1 und s in Phase 3 eine wichtige Rolle.

Die Gestalt des Pseudomoleküls am Ende von Phase 1 hängt in entscheidender Weise von der Wahl des Parameters r ab. Um hier grobe Fehler des Algorithmus zu vermeiden, führen wir diese Phase für verschiedene $r \in \{r_1, \ldots, r_k\}$ aus, mit der Erwartung, daß zumindest eines dieser r eine gute räumliche Approximation liefert. Gleichzeitig steigt jedoch auch die Laufzeit des Verfahrens mit der Größe dieses Intervals. Weiterhin sollte darauf geachtet werden, daß die beiden Moleküle, sollten sie von deutlich verschie-

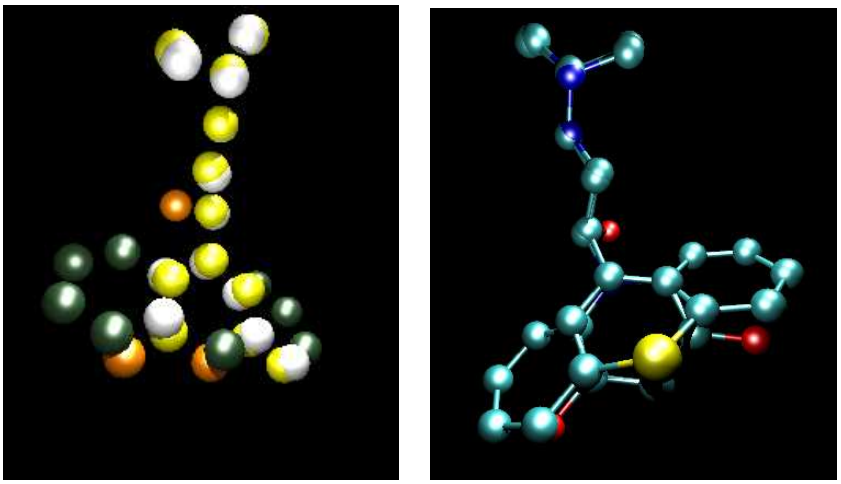


Tabelle 2.5: Endergebnis des Anwendungbeispiels: Links aus Sicht des Algorithmus; weiße und gelbe Atome sind einander zugeordnet, grüne und orange bleiben ohne Partner. Rechts in traditioneller Darstellung

dener Atomanzahl sein, in etwa in derselben Weise vergröbert werden, d.h. daß das Verhältnis von Pseudoatomen zu Ausgangsatomen jeweils ungefähr gleich bleiben sollte.

Etwas weniger sensibel reagiert der Algorithmus auf die Wahl von s in Phase 3. Jedoch auch hier hat es sich als hilfreich erwiesen, ein kleines Interval $s \in \{s_1, \ldots, s_l\}$ von verschiedenen Werten zu testen.

Es ist möglich, daß die exakte Lösung der artifiziellen Instanzen in Phase 2 gerade nicht die gewünschte Lage der Ausgangsmoleküle liefert, da wir aus zahlreichen Tests des exakten Verfahrens wissen, daß oft sehr viele verschiedene Zuordnungen existieren, die sich stark voneinander unterscheiden, jedoch fast den gleichen Score–Wert liefern. Um die Gefahr grober Fehler an dieser Stelle zu mindern, merken wir uns im Verlauf der Enumeration des exakten Verfahrens nicht nur die jeweils beste Lösung, sondern deren l.

Die Laufzeit des gesamten Verfahrens hängt in etwa linear von l und der Größe des s–Intervals ab, sie wächst jedoch superlinear mit größeren Werten für r.

In den Anwendungen (siehe 3.2) verwenden wir für Moleküle, die Kandidaten für Medikamentenwirkstoffe sind (und im Durchschnitt aus ca. 25 Nichtwasserstoffatomen bestehen), die Parameter $l = 40$, $r \in \{3, 4, 5\}$ und

$s \in \{3, 4\}$. Diese Wahl zeigte nach Auswertung zahlreicher Tests eine gute Balance zwischen Qualität und Laufzeitverhalten.

2.1.3.5 COSMOS–Webinterface

COSMOS (Comparing Small Molecules for Similarity) [HRT04] ist ein öffentlich zugängliches Webinterface, welches es dem Nutzer ermöglicht, Moleküle auf 3–dimensionale Ähnlichkeit mit dem vorausgehend beschriebenen Algorithmus zu vergleichen.

Nach Eingabe der Daten der Moleküle (als MDL-file)[5] ([DNH+92]) liefert COSMOS als Ergebnis den errechneten Score-Wert und die dazugehörige Überlagerungszuordnung. Parametereinstellungen kann der Nutzer frei wählen; die Koordinaten der überlagerten Moleküle können direkt übernommen werden.

Eine interaktive Visualisierung ermöglicht eine genaue Untersuchung des erhaltenen Ergebnisses.

Abbildung 2.1 soll einen Eindruck der Funktionalität von COSMOS liefern.

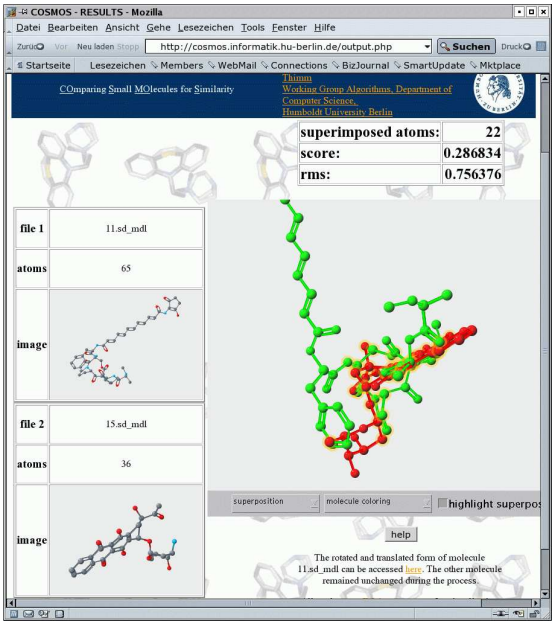


Abbildung 2.1: COSMOS-Webinterface: Ergebnis einer Überlagerung

[5] Andere gängige Formate wie z.B. SD, PDB o.ä. können mit Open Babel [bab] leicht konvertiert werden.

2.1.4 $3D$-Überlagerungsalgorithmus II

In diesem Kapitel beschreiben wir ein weiteres Verfahren, Moleküle auf 3–dimensionale Ähnlichkeit zu vergleichen. Es unterscheidet sich von den in den Abschnitten 2.1.2.2 und 2.1.3 beschriebenen Ansätzen in vielfältiger Weise:

Die Flexibilität der Moleküle wird nicht mehr dadurch behandelt, daß nacheinander alle möglichen Paare von Konformeren starr überlagert werden, sondern wir beziehen diese direkt in den Überlagerungsprozess mit ein, d.h. wir behandeln alle Konformere simultan.

Die zu optimierende Zielfunktion ist leicht verändert, d.h. strukturelle Eigenschaften der zu vergleichenden Moleküle werden als Eingabedaten mitbetrachtet. Die Ergebnisse des Algorithmus aus Abschnitt 2.1.3 in der Anwendung mit kleinen Molekülen (siehe 3.2) haben gezeigt, daß in der überwältigenden Mehrheit die optimalen bzw. sehr guten Überlagerungen gleichzeitig auch starke strukturelle Ähnlichkeit der den Molekülen zugeordneten Graphen aufwiesen, ohne daß dies der Algorithmus beachtet hätte. Wir kehren diesen Sachverhalt um und suchen also nur nach solchen Überlagerungen, in denen die Überlagerungszuordnung gleichzeitig eine Substrukturähnlichkeit auf Graphenebene induziert.

Eine Kombination dieser beiden Beobachtungen liefert schließlich einen neuen, sehr flexiblen und sehr schnellen Algorithmus, um Moleküle auf 3–dimensionale Ähnlichkeit zu vergleichen.

Im Folgenden soll zunächst auf die Änderung der Zielfunktion eingegangen werden. Danach beschreiben wir Methoden, geeignete zu überlagernde Substrukturen zu finden und beweisen einige Schranken, die dann schließlich dazu eingesetzt werden, die Anzahl der insgesamt zu enumerierenden Überlagerungen deutlich einzuschränken.

2.1.4.1 Zusammenhängende Überlagerungen

Um die Formulierungen im Folgenden präzise zu halten, definieren wir zuerst nochmals formal, was wir unter einem Bindungsgraphen verstehen wollen.

Definition 2.1.6 (Bindungsgraph) *Zu einem Molekül M betrachten wir den Bindungsgraphen $G_M = (V_M, E_M)$, wobei die Knotenmenge V_M mit der Menge der Nichtwasserstoffatome von M identifiziert wird und die Kanten den Atombindungen entsprechen, d.h*

$$E_M = \{\{a, b\} : \text{Atome a und b sind in } M \text{ durch eine Bindung verbunden}\}$$

Wie in der Einleitung dieses Kapitels erwähnt, wollen wir im hier beschriebenen Ansatz Substrukturinformationen mitbenutzen. Ähnlich wie in

den in 2.1.1.1 beschriebenen Verfahren suchen wir gemeinsame Untergraphen der zu vergleichenden Bindungsgraphen. Im Gegensatz zu den oben besprochenen Verfahren suchen wir hier nicht nur nach induzierten, sondern finden auch schwache, nicht induzierte Untergraphen. Wir verlangen zusätzlich, daß diese Untergraphen zusammenhängend sind.

Wir können die Überlagerungszuordnung f und den durch diese Zuordnung definierten Untergraphen identifizieren. In dieser Weise können wir dann auch von zusammenhängenden Überlagerungen sprechen.

Betrachten wir die Bindungsgraphen G_A und G_B zu zwei Molekülen A und B, so können wir in der folgenden Weise eine Relation $\sim$ definieren:

Definition 2.1.7 (Kantenrelation $\sim$) *Seien $G_A = (V_A, E_A)$ und $G_B = (V_B, E_B)$ zwei Bindungsgraphen. Falls für zwei Tupel $x = (x_1, x_2)$, $y = (y_1, y_2) \in V_A \times V_B$ gilt, daß $\{x_1, y_1\} \in E_A$ und $\{x_2, y_2\} \in E_B$, so schreiben wir $x \sim y$.*

Definition 2.1.8 (Zusammenhängende Überlagerung) *Sei f eine Überlagerungszuordnung. Wir bezeichnen den Graphen $G(f)$ mit der Knotenmenge $f = \{(a, b) \in V_A \times V_B : f(a) = b\}$ und der Kantenmenge $\{\{(a_1, b_1), (a_2, b_2)\} : (a_1, b_1) \sim (a_2, b_2)\}$ als den durch f implizierten Graphen. Ist $G(f)$ zusammenhängend, so nennen wir auch f zusammenhängend.*

Anschaulich ist der durch f implizierte Graph $G(f)$ nichts anderes als der kantenmaximale gemeinsame Subgraph von G_A und G_B auf $f(A)$ bzw. $f^{-1}(B)$.

Die Menge aller zusammenhängenden Überlagerungszuordnungen bezeichnen wir mit $\mathcal{F}_z(A, B)$.

Wir können nun unsere neue, modifizierte Zielfunktion angeben:

Definition 2.1.9 ($\boldsymbol{score_z(A, B)}$) *Sei $f \in \mathcal{F}_z(A, B)$ und M_f gegeben, so definieren wir den $score_z$ von f vermöge*

$$score_z(f) := \frac{|f|}{\min(n, m)} \exp(-rmsd^*(f))$$

Schließlich sei

$$score_z(A, B) := \max_{f \in \mathcal{F}_z(A,B)} score_z(f).$$

Obige Definition besagt einzig, daß wir die Menge der zulässigen Überlagerungen auf die Teilmenge der zusammenhängenden Überlagerungen einschränken.

2.1.4.2 Erzeugung zusammenhängender Überlagerungen

Wir beschreiben in diesem Paragraphen einen Algorithmus zur Erzeugung aller zusammenhängenden Überlagerungen. In 2.1.1.1 haben wir verschiedene Ansätze zur Lösung ähnlicher Probleme gesehen. Aus mehreren Gründen folgen wir nicht den vielerorts favorisierten cliquenbasierten Verfahren: Außer der Struktur der Bindungsgraphen wollen wir weiterhin keine zusätzliche Eingabeinformation betrachten, d.h. keine Einschränkung überlagerbarer Atome auf gleichen Atomtypen u.ä., keine Einschränkung der Kanten auf gleichen Bindungstyp usw. Die entstehenden Kompatibilitätsgraphen werden viel zu dicht, um darin effizient nach Cliquen suchen zu können.[6] Weiterhin werden nicht nur kantenmaximale gemeinsame Subgraphen gefunden. Da mehrfachzusammenhängende Überlagerungen durch verschiedene Subgraphen repräsentierbar sind, entsprechen sie deswegen auch mehreren Cliquen im Kompatibilitätsgraphen. Auf heuristische Verfahren zur teilweisen Lösung dieser angesprochenen Probleme, wie sie in 2.1.1.1 und der dort angegebenen Literatur beschrieben sind, wollen wir an dieser Stelle verzichten, um weiterhin exakte Lösungen garantieren zu können.

Im Folgenden beschreiben wir also einen direkten, effizienten Ansatz zur kompletten Enumeration aller kantenmaximalen zusammenhängenden Überlagerungen.

Ein einfaches Schema, alle zusammenhängenden Überlagerungen zu erzeugen, ist in Algorithmus 5 angegeben.

Algorithmus 5 : Zusammenhängende Überlagerungen

Input : $G_A = (V_A, E_A), G_B = (V_B, E_B)$
Output : Menge der zusammenhängenden Überlagerungen
$U_0 := \big\{\{(a,b)\} \ : \ a \in V_A, b \in V_B\big\}$
$i \leftarrow 1$
while $U_{i-1} \neq \emptyset$ **do**
    $U_i := \big\{u \cup \{x\} \ : \ u \in U_{i-1},\ x \in V_A \times V_B,\ \exists y \in u : x \sim y\big\}$
    $i \leftarrow i + 1$
return $\bigcup_{0 \leq j < i} U_j$

Die Korrektheit ist leicht einzusehen. Jedoch sind wir mit zwei Schwierigkeiten konfrontiert:

1. Die Bildung der Elemente U_i für $i \geq 1$ ist nicht eindeutig, denn zu jedem $f \in U_i$ existieren mindestens zwei $f' \in U_{i-1}$ mit $f' \subset f$. Jede

[6] Eine einfache Implementation des klassischen cliquenbasierten Verfahrens scheitert auf Instanzen mit mehr als 30 Knoten.

dieser kann mit dem Tupel y, $\{y\} = f \setminus f'$ zu f ergänzt werden. Es muß also bei der Implementation dieses Verfahrens darauf geachtet werden, daß keine Überlagerung mehrfach erzeugt wird.

2. In der in Algorithmus 5 angegebenen Form verlangt das Verfahren das explizite Speichern der U_i, was einen enormen Speicherbedarf bedeutet.

Im Folgenden beschreiben wir eine andere Herangehensweise, die diese beiden Probleme löst. Wir benötigen zuerst einige Definitionen.

Definition 2.1.10 ($con_f(u, v)$) *Sei f eine Überlagerung und u, $v \in f$. Sei*

$$con_f(u, v) := \{f' \subseteq f : \{u, v\} \subseteq f' \text{ und } f' \text{ ist zusammenhängend}\}.$$

Definition 2.1.11 (Distanz: $dist_f(u, v)$) *Die Distanz von u und v in f ist definiert als*

$$dist_f(u, v) := \begin{cases} \infty & \text{, falls } con_f(u, v) = \emptyset \\ \min_{f' \in con_f(u,v)} |f'| - 1 & \text{, sonst} \end{cases}$$

$dist_f(u, v)$ ist nichts anderes als die Anzahl der Kanten auf einem kürzesten Weg von u nach v in $G(f)$.

Nun gilt Folgendes:

Lemma 2.1.2 (Abstandslemma) *Sei $f \in \mathcal{F}(A, B)$ und $x \in f$. Dann gilt:*

$$dist_f(a, b) = \min(dist_{f \setminus x}(a, b), dist_f(a, x) + dist_f(x, b)),$$

wobei wir $\min(\infty, \infty) := \infty$ setzen.

Beweis: Wegen $con_{f \setminus x}(a, b) \subseteq con_f(a, b)$ gilt mit $con_f(a, b) = \emptyset$ auch $con_{f \setminus x}(a, b) = \emptyset$. Mit $u \in con_f(a, x)$ und $v \in con_f(x, b)$ gilt $u \cup v \in con_f(a, b)$, und somit ist mit $con_f(a, b)$ mindestens eine der beiden Mengen $con_f(a, x)$ oder $con_f(x, b)$ ebenfalls leer. Dies zeigt die Richtigkeit des Lemmas für $dist_f(a, b) = \infty$.

Sei nun also $con_f(a, b) \neq \emptyset$ und $f^* = \arg\ \min_{f' \in con_f(a,b)} |f'|$.

Falls $x \notin f^*$, so ist $f^* \in con_{f \setminus x}(a, b)$ und weiterhin minimales Element dieser Menge. Also folgt $dist_f(a, b) = dist_{f \setminus x}(a, b)$.

Ist nun aber $x \in f^*$, so gilt $|f^*| \leq dist_f(a, x) + dist_f(x, b)$, denn für alle g, h mit $g \in con_f(a, x)$ und $h \in con_f(x, b)$ ist $g \cup h \in con_f(a, b)$.

Gleichzeitig ist wegen der Minimalität von f^* $G(f^*)$ ein Pfad $P := v_1 = a, v_2, \ldots, v_i = x, \ldots, v_n = b$ von a über x nach b. Offensichtlich ist $\bigcup_{1 \leq j \leq i} v_i \in con_f(a, x)$ und $\bigcup_{i \leq j \leq n} v_i \in con_f(x, b)$ und somit $|f^*| \geq dist_f(a, x) + dist_f(x, b)$.

□

Wir benötigen noch die folgende Definition.

Definition 2.1.12 (Konsistenz) *Sei f eine Überlagerung. Eine Menge h von Tupeln, $h \in A \times B$, heiße konsistent zu f, falls $f \cup h$ eine bijektive Funktion ist.*

Der folgende nichtdeterministische Algorithmus 6 entscheidet zu $f \in M \subseteq A \times B$, ob f zusammenhängend ist. Aus dem Beweis der Korrektheit des Algorithmus (siehe Lemma 2.1.3) ergeben sich zwei wesentliche Tatsachen, die uns schließlich helfen werden, den gewünschten Algorithmus zur effizienten Enumeration aller zusammenhängenden Überlagerungen anzugeben.

Algorithmus 6 : Entscheide Zusammenhang

Input : - $M \subseteq V_A \times V_B$
- $f \in \mathcal{F}(A, B)$ mit $f \subseteq M$
- $v \in f$

1 $f' \leftarrow \{v\}$
2 $i \leftarrow 1$
3 **while** $f' \neq f$ **do**
4 $\quad W_i^* := \{w \in M \ : \ dist_{f'+w}(v, w) = i\}$
5 $\quad$ **if** $W_i^* = \emptyset$ **then**
6 $\quad\quad$ **reject**
7 $\quad$ wähle zu f' konsistente Teilmenge $W_i \subseteq W_i^*$
8 $\quad f' \leftarrow f \cup W_i$
9 $\quad i \leftarrow i + 1$
10 **accept**

Lemma 2.1.3 (Korrektheit von Algorithmus 6)
*Der Algorithmus **Entscheide Zusammenhang** arbeitet korrekt.*

Beweis: Wir zeigen zuerst, daß der Algorithmus terminiert. Dazu müssen wir nur die Programmabläufe betrachten, welche die Abbruchbedingung „$f' = f$“ (in Zeile 3) nicht erfüllen.

In jedem Durchlauf der **while**–Schleife werden Tupel aus W_i^* zu f' hinzugefügt, deren Distanz zu v größer ist als die Distanz der Elemente aus W_j^*, $j < i$, zu v war, als diese zur Auswahl standen. Mit Lemma 2.1.2 folgt, daß die Distanzen der zu f' hinzugenommenen Elemente zu v konstant bleiben, d.h. $W_j^* \cap W_k^* = \emptyset$, falls $j \neq k$. Da M endlich ist, muß jeder Ablauf, der nicht akzeptiert, ein i^* erreichen mit $W_{i^*}^* = \emptyset$.

Wir zeigen nun, daß es keinen akzeptierenden Programmablauf gibt, falls f nicht zusammenhängend ist. Nach Voraussetzung gibt es also ein Tupel $v' \in f$ mit $dist_f(v, v') = \infty$. Somit gilt für alle $f'' \subseteq f$: $con_{f''}(v, v') = \emptyset$. v'

kann aber nur dann in ein W_k aufgenommen werden, wenn $con_{f'}(v, v') \neq \emptyset$, wenn also ein $w \in W_j$, $j < k$ mit $w \notin f$ existiert. Da jedoch im Lauf des Algorithmus kein Element aus f' entfernt wird, wird die Abbruchbedingung „$f' = f$“ nie erfüllt sein und somit der Algorithmus nicht akzeptieren.

Schließlich zeigen wir, daß zu jeder zusammenhängenden Überlagerung $f \subseteq M$ mit $v \in f$ ein akzeptierender Programmablauf existiert.

Nehmen wir das Gegenteil an: Sei f eine betragsmäßig kleinste Überlagerung, die keinen akzeptierenden Programmablauf habe. Offensichtlich ist $f \supset \{v\}$. Sei $l := \max_{w \in f} dist_f(v, w)$ und $W := \{w \in f : dist_f(v, w) = l\}$. Aufgrund der Minimalität von f muß $f^* := f \setminus W$ akzeptiert werden.

Bezeichne $l' := \max_{w' \in f^*} dist_{f^*}(v, w')$ und $W' := \{w' \in f^* : dist_{f^*}(v, w') = l'\}$. Sei $\sigma = \langle W_0, W_1, \ldots, W_{l''} \rangle$ ein akzeptierender Programmablauf für f^*. Da $\bigcup_j W_j = f^*$ und die Distanz $dist$ eindeutig ist, folgt: $l'' = l'$ und $W_{l'} = W'$. Nun existiert für jedes $a \in W$ ein $b \in W'$ mit $a \sim b$ – nach Konstruktion von W' – jedoch kein $c \in f^* \setminus W'$ mit $b \sim c$, denn sonst wäre $dist_f(a, v) < l$.

Betrachten wir nun einen Programmablauf zur Eingabe f, dessen Suffix σ sei, d.h. alle Tupel aus W sind in W^*_{l+1} enthalten. Nun ist aber W konsistent zu $\bigcup_{i \leq l} W_i$, somit kann $W_{l+1} = W$ gewählt werden. Widerspruch zur Annahme. □

Zwei wichtige Tatsachen, die im Lauf des Beweises sichtbar geworden sind, können wir festhalten:

1. Die Distanzen aller zu f' hinzugenommenen Tupel zu v bleiben konstant.

2. In einem akzeptierenden Programmablauf entsprechen die W_i eindeutig den Klassen $f/dist_f(v, \cdot) := \{w \in f : dist_f(v, w) = i\}$.

Wir können also folgern:

Korollar 2.1.1 *Seien $M \subseteq A \times B$ und $v \in M$ beliebig, aber fest gewählt. Dann gibt es zu jeder zusammenhängenden Überlagerung $f \subseteq M$ mit $v \in f$ genau einen akzeptierenden Programmablauf von Algorithmus 6.*

Weiterhin gilt: Da jede Überlagerung f', welche der Algorithmus 6 erzeugen kann, potentiell zum Abbruch der **while**–Schleife in Zeile 3 führt und somit akzeptierbar ist, können wir wegen der Korrektheit von Algorithmus 6 folgern:

Korollar 2.1.2 *Seien $M \subseteq A \times B$ und $v \in M$ beliebig, aber fest gewählt. Bezeichne $\mathcal{W}$ die Menge aller Programmabläufe, welche der Algorithmus auf allen Eingaben f durchlaufen kann. Dann existiert eine Bijektion zwischen $\mathcal{W}$ und der Menge der zusammenhängenden Überlagerungen $f \subseteq M$ mit*

$v \in f$, wobei die einer Sequenz $\langle W_0, W_1, \dots \rangle \in \mathcal{W}$ zugeordnete Überlagerung unmittelbar durch die vereinigte Menge der Tupel $\bigcup_i W_i$ gegeben ist.

Diese Sequenzen $\mathcal{W}$ nun können aber einfach durch Rekursion enumeriert werden. Algorithmus 7 löst diese Aufgabe auf Eingabe $(M, v, \{v\})$.

Algorithmus 7 : Rekursive Aufzählung

Input : - $M \subseteq V_A \times V_B$
- $v \in M$
- zusammenhängende Überlagerung $f \subseteq M$ mit $v \in f$
- $i \leftarrow 1$

Output : Menge aller zusammenhängenden Überlagerungen f' mit $v \in f \subseteq f' \subseteq M$

```
1 R ← {f}
2 W* := {w ∈ M : dist_{f+w}(v, w) = i}
3 forall W ← zu f konsistente Teilmenge aus W* do
4     R ← R ∪ Rekursive Aufzählung(M, v, f ∪ W, i + 1)
5 return R
```

Schließlich können durch geeignete Unterteilung von $\mathcal{F}(A, B)$ alle zusammenhängenden Überlagerungen erzeugt werden. Wir definieren dazu eine strikte Ordnung auf $A \times B$ und erzeugen zuerst alle zusammenhängenden Überlagerungen, die das lexikographisch kleinste Tupel v_1 enthalten, dann alle jene, die das zweitkleinste Tupel v_2 enthalten, nicht aber v_1, usw. Diese Vorgehensweise ist in Algorithmus 8 beschrieben.

Algorithmus 8 : Erzeugung zusammenhängender Überlagerungen

Output : Alle zusammenhängenden Überlagerungen

```
1 U ← ∅
2 for i ∈ 1 ... nm do
3     U ← U ∪ Rekursive Aufzählung(A × B − ⋃_{j<i} {v_j}, v_i, {v_i})
4 return U
```

2.1.4.3 Schranken

Wir geben in diesem Kapitel eine Reihe von Schranken und ähnliche Techniken an, die es dann erlauben, den Enumerationsbaum des Verfahrens, das als Branch and Bound–Algorithmus implementiert ist, sehr effektiv zu beschneiden.

Wir beginnen mit zwei einfachen Schranken, die sich unmittelbar aus der Definition 2.1.9 der Score–Funktion ergeben:

Für jede Überlagerung f gilt:

$$score_z(f) \leq \frac{|f|}{\min(n,m)} \tag{2.8}$$

Für jede Überlagerung $f \supset f'$ gilt:

$$score_z(f) \leq \frac{|f|}{\min(n,m)} \exp\big(-\sqrt{\frac{|f'|}{|f|}} \cdot rmsd^*(f')\big) \tag{2.9}$$

Da die Berechnung des Score–Wertes wegen der darin benötigten Singulärwertzerlegung sehr zeitaufwendig ist, sind wir daran interessiert, eine Abschätzung für diesen Wert zu erhalten, der wesentlich schneller berechnet werden kann.

Seien a_1 und $a_2 \in A$ zwei beliebige Atome, die durch eine Überlagerung f auf $f(a_1) = b_1$ und $f(a_2) = b_2$ mit $b_1, b_2 \in B$, abgebildet werden. Sei M eine beliebige Bewegung von B und sei mit D die quadratische Distanz der Atome unter M bezeichnet,

$$D := |\underbrace{a_1 - M(b_1)}_{:=\Delta_1}|^2 + |\underbrace{a_2 - M(b_2)}_{:=\Delta_2}|^2.$$

Da M eine starre Bewegung ist, gilt folgende Nebenbedingung:

$$|M(b_1) - M(b_2)| = |b_1 - b_2| =: \beta.$$

Definieren wir noch $|a_1 - a_2| =: \alpha$, so erhalten wir mit der Dreiecksungleichung:

$$|\Delta_1| + |\Delta_2| \geq |\alpha - \beta|$$

bzw.

$$D = |\Delta_1|^2 + |\Delta_2|^2 \geq \frac{1}{2}|\alpha - \beta|^2 = \frac{1}{2}(\alpha - \beta)^2$$

Somit können wir folgendes Lemma formulieren:

Lemma 2.1.4 (Abschätzung für $score_z$) *Seien a_1, $a_2 \in A$ zwei Atome, die durch f auf $f(a_1) = b_1$ und $f(a_2) = b_2$ mit $b_1, b_2 \in B$ abgebildet werden, sowie M eine beliebige Bewegung von B. Dann gilt:*

$$|a_1 - M(b_1)|^2 + |a_2 - M(b_2)|^2 \geq \frac{1}{2}\big(|a_1 - a_2| - |b_1 - b_2|\big)^2$$

□

Mit Hilfe von Lemma 2.1.4 können wir nun eine einfache Abschätzung für $rmsd^*(f)$ angeben. Wir gehen hier davon aus, daß $|f|$ gerade ist, im anderen Fall müssen wir ein Tupel verwerfen.

Korollar 2.1.3 *Sei π eine beliebige Partition von $f^{-1}(B)$ in Teile zu je zwei Atomen, dann gilt mit Lemma 2.1.4:*

$$\begin{aligned} rmsd^*(f) &= \min_{M\in\mathcal{B}} \sqrt{\frac{1}{|f|}\sum_{(v_1,v_2)\in f} |v_1 - M(v_2)|^2} \\ &\geq \sqrt{\frac{1}{2|f|}\sum_{\{a_1,a_2\}\in\pi} \big(|a_1 - a_2| - |f(a_1) - f(a_2)|\big)^2} \end{aligned}$$

Korollar 2.1.3 besagt also, daß $rmsd^*(f)$ einer Überlagerung f umso größer ist, je mehr sich die Distanzen von jeweils aufeinander abgebildeten Atomen unterscheiden. Diese Beobachtung spielt nun eine Rolle, wenn wir alle Konformere eines Moleküls simultan betrachten wollen.

Einleitend stellen wir uns die Frage, ob alleine die Existenz einer großen Distanz

$$D_{ij} := ||a_i - a_j| - |b_i - b_j||$$

hinreichend sein kann für eine gute Abschätzung des resultierenden $rmsd^*$.

Benutzen wir Korollar 2.1.3, so folgt, daß

$$rmsd^*(f) \geq \sqrt{\frac{1}{2|f|}} \cdot D_{ij}.$$

Angenommen, wir wollen eine Überlagerung f erhalten, die einen Score–Wert von mindestens S hat, so bedeutet dies:

$$\begin{aligned} S \leq score_z(f) &= \frac{|f|}{\min(n,m)} \exp(-rmsd^*(f) \\ &\leq \frac{|f|}{\min(n,m)} \exp\Big(-\sqrt{\frac{1}{2|f|}} \cdot D_{ij}\Big). \end{aligned}$$

Soll also eine große Distanz D_{ij} dies unmöglich machen, so muß umgekehrt gelten:

$$D_{ij} \geq \sqrt{2|f|}\ln\Big(\frac{|f|}{S\min(n,m)}\Big) \tag{2.10}$$

Mit 2.8, 2.9, 2.1.3 und 2.10 haben wir nun vier Schranken beschrieben. Im folgenden Abschnitt zeigen wir, wie diese benutzt werden können, um effektiv

die Enumeration aller zulässigen Lösungen zu beschleunigen. Wir werden dann auch unterscheiden zwischen solchen Methoden, die weiterhin die global optimale Lösung garantieren, solchen, die dies nur tun, falls das Optimum genügend groß ist und Heuristiken, die für die praktische Anwendung ein außerordentlich schnelles Verfahren liefern, jedoch keine Optimalität mehr garantieren können.

2.1.4.4 Anwendung der Schranken

Wir verwenden die Bezeichnungen aus Algorithmus 7, dem Grundschema der Enumeration.

Überlagerungen mit Mindestgröße Die Schranken 2.8 und 2.9 zeigen einen einfachen Zusammenhang zwischen noch zu erreichendem Score–Wert und der Größe der Überlagerung f. Allgemein gilt, geben wir keine untere Schranke für den zu erreichenden Score–Wert an, daß wir stets nur noch solche Überlagerungen f betrachten müssen, für die gilt, daß

$$|f| \geq s_0 \cdot \min(n, m),$$

wobei s_0 der beste bisher berechnete Score–Wert sei. Dieser Wert s_0 kann also im Verlauf der Enumeration dynamisch angepaßt werden.

In praktisch relevanten Anwendungen sind wir meist nur an Überlagerungen interessiert, deren Score–Wert recht hoch ist, $score_z(f) \geq t$, mit z.B. $t \geq 0.7$. Hingegen ist es von geringem Interesse, ob sich zwei Moleküle nun „unähnlich“ oder „sehr unähnlich“ sind. Dieser Überlegung folgend können wir also dann den Wert s_0 von Beginn an mit $s_0 := t$ initialisieren. Die Exaktheit des Algorithmus leidet dann nur im unteren Score–Bereich, d.h. sollte ein Vergleich einen Score–Wert kleiner t liefern, so ist dieser nicht notwendig exakt, jedoch wissen wir sicher, daß der optimale Wert kleiner als t ist.

Diese Vorgehensweise können wir nun in Algorithmus 7 integrieren. Sei $M \subseteq A \times B$, so definieren wir

$$\begin{aligned} P_A(M) &:= \{a \in A : \exists (m_1, m_2) \in M : a = m_1\} \\ P_B(M) &:= \{b \in B : \exists (m_1, m_2) \in M : b = m_2\} \\ P(M) &:= \min(|P_A(M)|, |P_B(M)|). \end{aligned}$$

D.h. $P(M)$ ist eine obere Schranke für den Beitrag einer noch zu konstruierenden Überlagerung in M; somit kann eine Rekursion in Algorithmus 7 abgebrochen werden, falls

$$|f'| + P(M \setminus \bigcup_i W_i^*) \leq s_0 \text{ gilt.}$$

Diese Überlegung können wir noch präzisieren, wenn wir beachten, daß nicht nur die Anzahl der Elemente in P_A und P_B groß genug sein muß, sondern daß sich in den durch diese Mengen induzierten Untergraphen von G_A und G_B sogar entsprechend große zusammenhängende Subgraphen befinden müssen.

Beide Überlegungen lassen sich leicht und mit vergleichsweise kleinem rechnerischen Aufwand integrieren.

Berücksichtigung der Topologie Die spezielle Struktur der Bindungsgraphen ermöglicht folgende einfache Strategie zum Abschätzen der Anzahl der noch zu überlagernden Knoten: Betrachten wir den Graphen H, der entsteht, wenn wir die zu dem Tupeln $v_1, v_2, \ldots, v_k \in f$ gehörenden Knoten aus G_1 und G_2 identifizieren, genauso wie die durch f beschriebenen Kanten. Seien $C_1, C_2, \ldots, C_l$ die zusammenhängenden Komponenten von H, die entstehen, wenn wir die Kanten, die zu f gehören, löschen. Jede solche Komponente C_i besteht also aus c_1^i Knoten aus $G_1 \setminus f|_{G_1}$, c^i Knoten, die zu f gehören und c_2^i aus $G_2 \setminus f|_{G_2}$. Da die Erweiterung von f stets zusammenhängend sein soll, können also nur Knoten innerhalb dieser Komponenten aufeinander abgebildet werden. Somit ist f um maximal $\sum_{j=1}^{l} \min(c_1^j, c_2^j)$ Knoten erweiterbar.

Überlagerung mit beschränkter Maximaldistanz D_{ij} Die Schranken 2.1.3 und 2.10 sind ebenfalls verlustfrei, d.h. wenden wir sie in der oben formulierten Form an, erhalten wir stets garantiert die optimale Lösung zurück. Gleichzeitig gilt aber auch, daß eine starke Reduktion der Größe des Enumerationsbaumes in dieser Weise nicht möglich ist; auch sind die Methoden, insbesondere 2.1.3 recht zeitaufwendig, da sie stets für alle Paare von Konformeren ausgeführt werden müssen.

Schranke 2.10 bietet eine gute Möglichkeit, von der Notwendigkeit des paarweisen Vergleichs der Konformere wegzukommen.

In der Formulierung von Schranke 2.10 ist von Konformeren nicht die Rede, d.h. sie gilt für je ein Konformer von A und B. Effektiver algorithmisch überprüfbar wird sie, wenn wir folgende Vorarbeiten leisten:

Definition 2.1.13 (Distanzmatrizen) *Sei A ein Molekül mit der Atommenge $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ und den Konformeren $\mathcal{K}_A = \{K_1, K_2, \ldots, K_l\}$ mit $K_i : \mathcal{A} \to \mathbb{R}^3$. Wir definieren die Matrizen $U(A)$ und $O(A)$ vermöge*

$$U(A) \in \mathbb{R}^{n \times n} \text{ mit } U(A)_{ij} := \min_{1 \leq r \leq l} |K_r(a_i) - K_r(a_j)|$$

und

$$O(A) \in \mathbb{R}^{n\times n} \text{ mit } O(A)_{ij} := \max_{1\le r\le l} |K_r(a_i) - K_r(a_j)|.$$

Zu einem Paar von Atomen $a_i, a_j \in A$ und $\delta \in \mathbb{R}^+$ sei mit $I(A,\delta)_{ij}$ das Interval $[U(A)_{ij} - \delta, O(A)_{ij} + \delta]$ bezeichnet.

Für den exakten Gebrauch der Schranken 2.1.3 und 2.10 ist es nun möglich, die Werte D_{ij} mit einem Blick in die Einträge der Matrizen $U(A)$, $U(B)$, $O(A)$ und $O(B)$ global über alle Konformationspaare abzuschätzen. Zusätzlich können, mit größerem Testaufwand, für einen Ast der Enumeration je Teilmengen der Konformationsmengen $\mathcal{K}_A$ und $\mathcal{K}_B$ ausgeschlossen werden und müssen so in tieferen Ebenen nicht mehr betrachtet werden. In diesem Fall kann auch ein Aktualisieren der Matrizen gemäß der reduzierten Konformationsmengen sinnvoll sein.

Eine wirklich große Reduktion des Enumerationsbaumes für praktisch relevante Anwendungen erhalten wir mit folgender Überlegung: Die Schranke 2.10 ist für den allgemeinen Fall nicht stark genug, jedoch können wir fordern, daß wir nur solche Überlagerungen f zulassen wollen, für die gilt, daß $D_{ij} \le \beta$ für ein festes β eingehalten wird.

Es lassen sich theoretisch Konstellationen bilden, in denen eine optimale Überlagerung f Paare zugeordnet hat, die für D_{ij} große Werte ergeben. Aus Sicht der Anwendung ist dies allerdings sehr unwahrscheinlich, und im Falle des Auftretens eher unerwünscht.

Sei f eine Überlagerung. Existieren zwei Tupel (v_1, v_2) und $(w_1, w_2) \in f$ mit

$$I(A,\beta)_{v_1w_1} \cap I(B,\beta)_{v_2w_2} = \emptyset,$$

so folgt daraus, daß sich die Abstände der überlagerten Atompaare in allem Paaren von Konformeren um mindestens 2β unterscheiden.

Wir können also die Enumeration der zusammenhängenden Überlagerungen in Algorithmus 7 in der Weise modifizieren, daß in der dritten Zeile für jedes neu aufgenommene Tupel (u_1, u_2) getestet wird, ob obiger Intervallschnitt leer ist für alle $(w_1, w_2) \in f$. Der Rechenaufwand ist im Vergleich zum Ertrag verschwindent.

Bemerkungen Die in den beiden vorhergehenden Paragraphen benannten Anwendungen der Schranken sind nicht die einzigen, die die aktuelle Implementation des Algorithmus enthält, jedoch sind es die wesentlichen. Eine Vielzahl von Modifikationen und Erweiterungen ist denkbar, z.B. die bereits angedeutete Auffrischung der Abstandsmatrizen auf einzelnen Ästen der Rekursion. Auch sind einige Heuristiken implementiert, die Wahl der Partition π in 2.1.3 zu optimieren. Weiterhin kann man daran denken, globale,

schwerpunktbasierte Schranken, ähnlich der in Lemma 2.1.1 beschriebenen, einzubauen.

Alle diese Maßnahmen hätten und haben eine weitere, z.T. deutliche Reduktion des Enumerationsbaumes zur Folge. Jedoch sind die Kosten dafür, also der Aufwand an Rechenzeit, in verschiedenen Anwendungen höher als die Ersparnis.

Wichtig ist, daß der Anwender durch die Wahl der beiden entscheidenden Parameter s_0 und β entscheiden kann, ob, in welchem Wertebereich und mit welchem Risiko er auf eine Garantie der Optimalität der Ergebnisse verzichten will.

2.1.4.5 Patent

Die in den Abschnitten 2.1.4.1 – 2.1.4.4 beschriebenen Techniken bilden die Basis der Patentanmeldung [HTZ05] „Verfahren und Vorrichtung zum computergestützten Auffinden von ähnlichen Molekülen “, die am 24.06.2005 beim Deutschen Patent– und Markenamt zur Patentierung eingereicht wurde.

2.2 Clustern

2.2.1 Überblick

Die Aufgabe, eine Menge von Objekten so in Gruppen zu unterteilen, daß Objekte innerhalb einer Gruppe sich ähnlich sind und unähnliche Objekte in verschiedenen Gruppen liegen, wird als Clustern bezeichnet.

Je nach Blickwinkel und Anwendung kann diese Definition präzisiert werden und liefert so zahlreiche, verschiedene Aufgabenstellungen; ebenso zahlreich sind die Anwendungsgebiete. Viele verschiedene Ansätze zur Lösung von Clusterproblemen sind bekannt (siehe z.B. [JMF99]). Wir wollen hier nicht versuchen, einen kompletten Überlick zu geben – dies ist im Rahmen der vorliegenden Arbeit nicht möglich – sondern beschränken uns darauf, einige verschiedene Methoden kurz zu skizzieren.

Die Definition der Ähnlichkeit ist eine Frage, die vor dem Clustern beantwortet werden muß. Daß es zahlreiche verschiedene Antworten auf diese Frage gibt, die im Wesentlichen von der betrachteten Anwendung abhängen, sei an dieser Stelle bemerkt. In 2.2.3.1 sehen wir, daß in konkreten Situationen nicht immer nur eine Möglichkeit der Ähnlichkeitsdefinition existiert. In 3.3.2 wird aufgezeigt, wie diese Unterschiede – bei gleichen Clustermethoden – die Ergebnisse beeinflussen können.

Ohne Anspruch auf Vollständigkeit, können folgende Klassen von Clusterverfahren unterschieden werden:

Hierarchische Methoden verschmelzen sukzessive immer die beiden nächsten Cluster zu einem neuen (z.B. [Kin67]). In dieser Weise entsteht eine Baumstruktur von Clustern, ein sogenanntes Dendogramm. Nun kann die Nähe zwischen zwei Clustern unterschiedlich definiert werden, als kleinster (single link), größter (complete link) oder durchschnittlicher (average link) Abstand zweier Punkte der verschiedenen Cluster. Auch die umgekehrte Variante tritt auf, d.h. man startet mit einem Cluster – der gesamten Menge der Objekte – teilt diese in zwei Teile und fährt analog fort.

Partitionsmethoden versuchen von Beginn an, die Menge der Objekte in eine gegebene Anzahl von k Clustern zu unterteilen, die eine gewisse, meist lokale Optimalitätsbedingung erfüllt. Bekanntester Vertreter dieser Familie ist vielleicht der K–means Algorithmus [McQ67]. In diesem Ansatz wird versucht, die Zielfunktion

$$E = \sum_{j=1}^{k} \sum_{x \in C_j} d(x, m_j)$$

zu minimieren, wobei die m_i jeweils die Zentren der Cluster C_i bezeichnen und $d(x, m_i)$ die Distanz eines Elementes x zu m_i mißt. Der Algorithmus startet mit der Wahl von k Clusterzentren m_i, dann wird jedem Objekt x ein Cluster zugewiesen, nämlich jenes, zu dem es den kleinsten Abstand hat. Iterativ werden nun neue Zentren berechnet, bis diese sich nicht mehr ändern.

Spektrale Methoden nutzen Techniken der linearen Algebra, die Singulärwertzerlegung der Datenmatrix $A \in \mathbb{R}^{n \times d}$ (n Elemente in $\mathbb{R}^d$), um eine Approximation D für A von kleinem Rang k zu erhalten. Mit Hilfe dieser läßt sich dann eine Clusterung der Zeilen von A erhalten (siehe z.B. [KVV04], [DFK+99]).

Sonstige: Verschiedene andere Methoden finden Anwendung: Künstliche Neuronale Netze [RSS+02], deterministische [HB97] und randomisierte [KD89] Suchverfahren, Genetische Algorithmen [RB79] und viele andere mehr.

In diesem Zusammenhang soll bemerkt werdem, daß sich die Bewertung der Qualität einer Clusterung als oft recht schwierig erweist. In vielen Fällen, wie auch in den von uns in dieser Arbeit vorgestellten, entscheidet die Interpretation der Daten aus dem Blickwinkel der Anwendung mit über die Güte der Clusterung.

Auch aus theoretischer Perspektive erweisen sich solche Fragen als interessant. Ein axiomatischer Ansatz ist der Folgende: Sei f eine Clusterfunktion, also eine Funktion, die als Eingabe die Menge S der Objekte und paarweise Distanzen dieser erhält und eine Partition von S als Ausgabe liefert. Versuche, allgemeine Bedingungen an eine Clusterfunktionen zu stellen, die in jedem Fall wünschenswert erscheinen und erfüllt sein sollen, liefern gelegentlich überraschende Resultate. Soll f z.B. invariant unter Skalierung der Daten sein, prinzipiell jede Partition als Lösung liefern können, und in der Weise konsistent sein, daß die Clusterung erhalten bleibt, wenn die Distanzfunktion so geändert wird, daß Objekte innerhalb von Clustern sich nur annähern und solche in verschiedenen Clustern ihre Distanz erhöhen, so wird in [Kle02] bewiesen, daß eine solche Clusterfunktion f nicht existieren kann.

Biclustern Ein allgemeineres Problem, das sogenannte Biclustern, lehnt sich noch wesentlich stärker an die Anwendung und ist hauptsächlich im Zusammenhang mit der Auswertung von Genexpressionsdaten entstanden.

Abstrakt stellt sich die Aufgabe wie folgt: Gegeben ist eine Menge O von n Objekten und eine Menge B von m Bedingungen. Für jedes Objekt i liegt zu jeder Bedingung j ein Datenwert a_{ij} vor. Aufgabe ist es nun, eine Teilmenge

$I \subset O$ zu finden, die sich unter einer Teilmenge $J \subset B$ der Bedingungen in einer geeignet definierten Weise ähnlich verhält.

Da, wie bereits erwähnt, fast alle Ansätze zur Lösung dieses Problems aus ganz konkreten Anwendungen entstanden sind, sind die verschiedenen vorgeschlagenen Algorithmen sehr unterschiedlich und kaum vergleichbar, da oft die Auswertung nur im Hinblick auf die betrachtete Anwendung erfolgt und die Clusterung mathematisch meist nicht zu bewerten ist. Eine mathematische Theorie des Biclusterns hat sich noch nicht herausgebildet.

Einen Überlick über die verschiedenen vorgeschlagen Methoden findet man z.B. in [MO04] oder [TS05].

2.2.2 Clustern molekularer Strukturen

In diesem Abschnitt beschreiben wir zwei Methoden, die Elemente einer Datenbank molekularer Strukturen in einer Weise anzuordnen, die es möglich macht, diese danach effizient zu durchsuchen.

Bei der Datenbank DIP (The Dictionary of Interfaces in Proteins [PGF98]) handelt es sich um eine Kollektion von 3–dimensionalen Strukturen von interagierenden Teilen von Proteinen. Genauer wollen wir diese in 3.1 beschreiben.

Die betrachtete Fragestellung ist die Folgende: Zu einer neuen Struktur N wollen wir alle zu ihr ähnlichen Strukturen n_i aus der Datenbank finden. Dies soll mit möglichst wenigen Anfragen an die Datenbank gelingen. Der naive Ansatz, die neue Struktur gegen alle Elemente der Datenbank zu testen, ist zu zeitaufwendig, da jedesmal linear viele Anfragen notwendig sind, was bei einer Anzahl von ca. 10^7 Datenbankeinträgen zu kostspielig ist.

Die Ähnlichkeitsfunktion, mit der die Strukturen verglichen werden, ist dieselbe wie in Definition 2.1.5. Jedoch sind die Werte dieser Funktion hier anders zu deuten als beim Vergleich kleiner Moleküle. Schon kleinere Score–Werte signalisieren einen gewissen gesuchten Grad an Ähnlichkeit. Eine erste Aufgabe besteht nun darin, die Datenbank auf stukturelle Eigenschaften hin zu untersuchen: Ab welchem Score–Wert kann von Ähnlichkeit gesprochen werden? Gibt es verschiedene Grade von Ähnlichkeit, und wie lassen sich diese erklären?

Positive Anworten darauf sollten es dann möglich machen, die Ausgangsfragestellung, bezogen auf das Sortieren und Suchen, besser lösen zu können.

2.2.2.1 Analyse der Eingabedaten

In der hier betrachteten Anwendung fand zum Vergleich der Strukturen ein bisher noch nicht erwähnter Algorithmus, der in [PGF98] erstmals beschrieben wird, seine Anwendung. Aufgrund der Vorgehensweise dieses Algorith-

mus, der nicht zum Vergleich von Strukturen von deutlich verschiedener Größe geeignet ist, haben wir die Strukturen der Datenbank in Klassen von ungefähr gleicher Größe unterteilt. Alle folgenden Aussagen über Struktureigenschaften von Graphen, die nun aus den so gebildeten Klassen von molekularen Strukturen hervorgehen, gelten also immer für eine solche Klasse. Genaueres dazu findet sich in 3.1.1.

Schwellenwertgraph Zu einer Menge M von n molekularen Strukturen bilden wir den vollständigen gewichteten Ähnlichkeitsgraphen $G(M)$ in der folgenden Weise: Die Knoten von $G(M)$ entsprechen den Strukturen. Zwei Knoten u und v sind mit einer Kante verbunden, deren Gewicht $w(\{u, v\})$, dem paarweisen Score–Wert, $score(u, v)$, entspricht.

Zu jedem $t \in \mathbb{R}$, $0 < t < 1$, können wir nun den ungewichteten Schwellenwertgraphen $G_t(M)$ definieren: $V(G_t(M)) = V(G(M))$, $E(G_t(M)) = \{e \in E(G(M)) : w(e) \geq t\}$.

Wir vergleichen nun strukturelle Eigenschaften von $G_t(M)$ und einem zufälligen Graphen H_t. Die Evolution zufälliger Graphen ist ein gutstudiertes Gebiet (siehe z.B. [Bol01]). Sollten die Daten eigene, von zufälligen Daten abweichende Struktur enthalten, so müßte dies in der unterschiedlichen Evolution von $G_t(M)$ und H_t erkennbar sein. Wir bilden H_t, indem wir die Einträge der Adjazenzmatrix von $G_t(M)$ zufällig permutieren, d.h. die Verteilung der Kantengewichte von H_t ist identisch mit der von $G_t(M)$. Aus der Theorie der zufälligen Graphen ist bekannt, daß die unten genannten untersuchten Eigenschaften, ist der Graph in der Tat zufällig, stabil sind unter Permutation der Kanten. Dies bedeutet, daß erkennbare Unterschiede in den beobachteten Parametern von $G_t(M)$ und H_t auf „versteckte“ Struktur von $G_t(M)$ schließen lassen.

Wir simulieren die Evolution von $G_t(M)$ und H_t für t, von 0 nach 1 wachsend. Die Parameter, auf die wir die Graphen untersuchen, sind

- Anzahl der zusammenhängenden Komponenten
- Durchschnittsgröße einer Komponente
- Anzahl isolierter Knoten
- Größe der größten Komponente

Den Ergebnissen vorgreifend, die in 3.1.2.1 genauer dargestellt sind, gilt: Der einzige Parameter, bezüglich dessen sich $G_t(M)$ und H_t signifikant unterscheiden, ist die Größe der größten Komponente. Gleichzeitig zeigt sich ein deutlicher Zusammenhang der Evolution dieses Parameters zur Gesamtverteilung der berechneten Score–Werte.

Es zeigen sich im Wesentlichen drei Bereiche von verschiedenen Ähnlichkeiten:

- Unähnlichkeit oder „Rauschen“
- „zufällige“ Ähnlichkeit, d.h. solche, die sich nicht aus der Sequenzähnlichkeit der Proteinsubstrukturen erklären läßt
- starke Ähnlichkeit, induziert zumeist durch Sequenzstrukturähnlichkeit

Diese drei Bereiche sind sichtbar getrennt, einerseits in der veranschaulichten Verteilung der gemessenen Ähnlichkeitswerte; andererseits finden sich die signifikanten Unterschiede der Größe der größten Komponente von $G_t(M)$ im Vergleich zu H_t genau an den Stellen, die diese Bereiche trennen (siehe 3.1.2.1).

Diese Einsichten in die Struktur der Daten läßt es sinnvoll erscheinen, die Datenstrukturen zum Ordnen und die Algorithmen zum Suchen in der Datenbank danach auszurichten.

2.2.2.2 Hierarchischer Ansatz

Für den dritten Bereich der sehr hohen Ähnlichkeiten zeigt ein hierarchischer Ansatz die beste Güte. Angenommen, wir sind nur an solchen $n_i \in M$ interessiert, deren Score–Wert im Vergleich zu einer neuen Struktur N mindestens t beträgt, wobei t vergleichsweise hoch ist. Der benutzte Wert für t wird den Ergebnissen von 2.2.2.1 entnommen. Wie in 3.1.2.1 gezeigt wird, hat der Graph $G_t(M)$ in diesen Bereichen für t eine Struktur, die den folgenden Zugang rechtfertigt. Sei $I := \{i \in \{1, 2, \ldots, n\} : \exists\{n_i, n_j\} : w(\{n_i, n_j\}) \geq t\}$. Wir betrachten dann nur den von I induzierten Graphen $G[I]$. Die dazugehörigen Daten werden dann in Form eines binären Baumes organisiert.

Ein n_i wird ausgewählt, alle übrigen n_j, $j \neq i$ mit n_i verglichen und gemäß Score–Wert sortiert. Nun wird ein sogenannter kritischer Wert s berechnet, der folgende Eigenschaften besitzt:

1. Die Mengen $A := \{n_k : score(n_i, n_k) \leq s, k \neq i\}$ und $B := \{n_j : score(n_i, n_j) > s, j \neq i\}$ sollen etwa gleich groß sein.
2. Die Anzahl der Kanten zwischen A und B in $G[I]$ soll so gering wie möglich sein.

Kann ein solcher Wert gefunden werden, so repräsentiert das Tripel (I, n_i, s_{n_i}) die Wurzel des Baumes, A bildet den linken Ast, B den rechten. Die Mengen A und B werden dann analog rekursiv unterteilt, solange die beiden Bedingungen innerhalb der vorgegebenen Toleranzen erfüllt sind.

Der in dieser Weise aufgebaute Baum kann nun sehr effizient durchsucht werden. Ist eine unbekannte Struktur N gegeben und wir suchen alle $n_i \in I$, deren Ähnlichkeit zu N groß ist, so gehen wir wie folgt vor:

Startend bei der Wurzel, vergleichen wir jeweils das Element N mit dem dem Knoten im Baum zugeordneten Element n_j. Ist der berechnete Score–Wert $score(N, n_j)$ kleiner als der in diesem Baumknoten gespeicherte kritische Wert s_{n_j}, so wird die Suche im linken Ast fortgesetzt, im anderen Fall im rechten Ast. Immer wenn bei einem solchen Vergleich ein Score–Wert $score(N, n_j) \geq t$ ausgegeben wird, so notieren wir n_j als ähnlich zu N. Die Anzahl der hier benötigten Vergleiche ist linear in der Tiefe des Baumes, also im besten Fall von der Größenordnung $\log_2 |I|$.

Bemerkungen Dieser Ansatz ist also darauf spezialisiert, sehr hohe Treffer, also Elemente die zu N sehr ähnlich sind, zu finden. Wir nutzen hier die Tatsache, daß die in den Daten befindlichen Klassen mit paarweise sehr hoher Ähnlichkeit nur sehr wenige sind, innerhalb dieser Klassen jedoch eine starke Transitivität bezüglich des Score–Wertes zu sehen ist. Mit anderen Worten: Wir erwarten, daß diejenigen neuen Elemente, die zu einem n_i sehr ähnlich sind, genauso in eine dieser wenigen Klassen fallen.

2.2.2.3 Dominating–Set–Ansatz

Betrachten wir nun den interessanten mittleren Bereich. Hier finden sich Ähnlichkeiten, die nicht mit Sequenzähnlichkeit zu erklären sind. Wiederum nach genauer Analyse der Struktur der Daten hat sich folgender Ansatz bewährt:

Wir verwenden wie in 2.2.2.2 nicht alle Kanten von $G(M)$, sondern nur jene, die ein gewisses Mindestgewicht t überschreiten, wobei nun t im Vergleich zu 2.2.2.2 wesentlich kleiner ist, und somit die Anzahl der Kanten, die wir mitberücksichtigen, wesentlich größer. Wieder dienen die Ergebnisse aus 2.2.2.1 als Auswahlkriterium für die Wahl von t.

Sei also erneut $I := \{i \in \{1, 2, \ldots, n\} : \exists\{n_i, n_j\} : w(\{n_i, n_j\}) \geq t\}$. In $G[I]$ suchen wir nun – mit Hilfe einer gewissen Greedy–Heuristik – eine Dominierende Menge, also eine Menge $D \subset V(G[I])$, für die gilt, daß alle Knoten n_i in $V(G[I])$ mit mindestens einem $d_i \in D$ benachbart sind. Wir bilden die Dominierende Menge D auf die folgende Weise:

1. Sortiere die Knoten $n_i \in V(G[I])$ absteigend nach durchschnittlichem Gewicht der Kanten zu ihren Nachbarn. Setze $D := \emptyset$

2. Füge solange Knoten n_i zu D hinzu – in der Reihefolge aus Schritt 1 – bis alle $n_j \in V(G[I])$ durch D überdeckt sind.

Die Knoten $d_i \in D$ bilden also eine Menge von Repräsentanten. Die durch diese Repräsentanten vertretenen Mengen sind i.a. nicht disjunkt, sondern weisen starke Überschneidungen auf.

Starten wir nun die Suche nach Elementen $n_i \in V(G[I])$, die ähnlich einem neuen Element N sind, so gehen wir wie folgt vor:

- Für alle Repräsentanten $d_i \in D$
  - Vergleiche N mit d_i
  - Falls $score(N, d_i) > t$: Vergleiche N mit allen Nachbarn von d_i

Bemerkungen Diese Basisversion der Suche kann nun mit zwei Parametern gesteuert werden, je nachdem, ob wir an einer wirklich sehr geringen Anzahl von benötigten Vergleichen interessiert sind und dabei in Kauf nehmen, daß wir einen gewissen Anteil an Treffern (falsch negative) nicht finden, oder ob eine sehr hohe Trefferquote Priorität hat, und wir dafür eine etwas größere Anzahl Anfragen akzeptieren. Der eine Parameter T gibt an, wieviele Treffer T mindestens gefunden werden sollen. Der zweite Parameter Q steuert, nach wievielen erfolglosen Anfragen für ein Element N die Suche abgebrochen wird. Ist also z.B. $T = 1$ und $Q = 20$, so wird die Suche wie oben beschrieben, solange fortgesetzt, bis mindestens ein n_i gefunden wurde, das ähnlich zu N ist. Danach kann es vorkommen, daß „lange“ nichts gefunden wird, d.h. eine Reihe von Vergleichen mit Repräsentanten d_i (bzw. mit Nachbarn n_i eines Treffers d_i) liefert immer einen Score–Wert $score(N, d_i) < t$ (bzw. $score(N, n_i) < t$). Nach 20 derartigen Anfragen stoppen wir dann die Suche.

Diese Vorgehensweise wird im Wesentlichen gerechtfertigt durch die Sortierung der Elemente $d_i \in D$. Wir erwarten die überwiegende Anzahl der Treffer in den ersten Repräsentantenmengen. Diese sind meist groß im Vergleich zu denen, die weit hinten in der Reihenfolge stehen; auch ist die durch die darin befindlichen Kanten beschriebene Ähnlichkeit der Elemente wesentlich höher. In dieser Weise kann die in gewissem Maß zu findende Transitivität der Score–Werte so gut wie möglich ausgenutzt werden.

2.2.3 Clustern nach Wirkungsähnlichkeit

In den bisherigen Betrachtungen haben wir stets Moleküle oder molekulare Strukturen auf ihre 3–dimensionale Ähnlichkeit untersucht. Hier nun beschreiben wir eine andere Möglichkeit, Ähnlichkeit von Molekülen zu definieren, nämlich nach ihrer experimentell gemessenen Aktivität in medizinischen Versuchsreihen.

Wir betrachten eine Menge O von n Objekten, wobei jedem dieser Objekte o_i eine Reihe $A_i = \{a_{i1}, a_{i2}, \ldots, a_{im}\}$ von Meßwerten zugeordnet ist. Bei den Objekten handelt es sich um potentielle Wirkstoffkandidaten, also um Moleküle. Die Meßwerte sind gemessene Wachstumshemmungsfähigkeiten der Substanzen bezüglich einer Menge von Krebszellen. Weitere Details über die betrachteten Daten und über die durchgeführten Experimente finden sich in 3.3.1.

Ziel des Folgenden ist es, einerseits die Objekte (Substanzen) gemäß Wirkungsähnlichkeit zu clustern, andererseits dann, diese Wirkungsähnlichkeit mit der berechneten $3D$-Ähnlichkeit zu vergleichen.

Die Frage nach dem Clustern der Daten gemäß Wirkungsähnlichkeit wurde bereits mehrfach bearbeitet. In [PSH+89] finden sich Ansätze, die Meßdaten zu veranschaulichen und erste Vergleichsalgorithmen werden vorgeschlagen. In [RSS+02] wird eine große Teilmenge der Substanzen mit Hilfe von heuristischen Methoden (SOMs, self–organizing maps [Koh95]) komplett klassifiziert.

Oft sind nur sehr spezielle Klassen von Molekülen, über die bereits andere Untersuchungen vorliegen, Untersuchungsgegenstand von Clusteransätzen. Eine genauere Unterscheidungen in Unterklassen ist das Ziel [SMF+98], [SFM+98].

Andere Arbeiten versuchen genauere Kenntnis über spezielle Unterklassen von Substanzen zu gewinnen, indem sie Wirkungsdaten mit weiteren Informationen, wie z.B. strukturellen Eigenschaften der Moleküle, anreichern [WMO+97].

Zwei Fragen stellen sich: Wie definieren wir Ähnlichkeit von zwei Substanzen anhand von Wirkungsdaten? Wie clustern wir die Daten? In den folgenden Abschnitten gehen wir diesen Fragen nach.

2.2.3.1 Definition der Ähnlichkeit von Wirkungsdaten

Bevor wir sinnvolle Ähnlichkeitsdefinitionen von Wirkungsdaten definieren können, müssen wir zunächst die vorhandenen Daten inhaltlich interpretieren.

Abstrakt liegen uns die Daten als eine Matrix $A \in \mathbb{R}^{n \times m}$ vor, wobei sich in der i–ten Zeile der Matrix, die einer Substanz o_i zugeordnet ist, m Zahlen a_{ij}, $1 \leq j \leq m$, befinden. a_{ij} gibt den Logarithmus der molaren Konzentration der Substanz o_i an, die notwendig ist, das Wachstum einer Krebszelle j um die Hälfte zu hemmen.[7]

[7] Dies ist eine idealisierte Darstellung. Da es sich um echte Meßdaten handelt, sind die tatsächlichen Daten einerseits z.T. offensichtlich fehlerhaft, andererseits fehlen viele Meßdaten, d.h. die Matrix A hat „Löcher“.

Ein Vergleich von zwei Substanzen entspricht also dem Vergleich zweier Zeilen der Matrix A. Das Interesse der Anwendung liegt nicht in der direkten Übereinstimmung einzelner Datenpunkte, auch nicht in der überwiegenden Gleichheit zweier Zeilen. Vielmehr interessiert man sich für ähnliche sogenannte Wirkungsprofile, d.h. Zeilen der Matrix, die in geeigneter Weise korreliert sind. Z.B. läge eine sehr gute Korrelation vor, wäre eine Menge von Zeilen nur um einen additiven Betrag verschoben. Dies signalisierte, daß die zugehörigen Substanzen ein im Wesentlichen gleiches Wirkspektrum hätten. Allein die benötigten Konzentrationen wären um einen konstanten Faktor verschieden, also z.B. benötigte man von der einen Substanz immer lediglich die Hälfte der anderen, dies jedoch gleichzeitig für alle Krebszellen.

2.2.3.1.1 Pearson–Korrelationskoeffizient Der oben angesprochene Sachverhalt ist nun in verschiedener Weise behandelt worden. Zumeist werden die Ausgangsdaten zuerst einer Normalisierung unterzogen. Da wie oben angedeutet additive Verschiebungen der Zeilen zugelassen werden sollen, werden also die Einträge einer Zeile um ihren Mittelwert verschoben. Meist wird dann auch noch durch die Standardabweichung geteilt, um die verschiedenen Sensitivitäten der Krebszellen abzumindern. Konkret wird die Normalisierung

$$z_{ij} := \frac{a_{ij} - \bar{a_i}}{s_i} \text{ vorgenommen,}$$

wobei

$$\bar{a_i} := \frac{1}{n} \sum_{j=1}^{n} a_{ij} \text{ den Mittelwert bezeichnet}$$

und

$$s_i := \sqrt{\sum_{j=1}^{n} \frac{(a_{ij} - \bar{a_i})^2}{n}} \text{ die Standardabweichnung.}$$

In [RSS+02] werden die so normalisierten Daten direkt als Punkte in $\mathbb{R}^m$ behandelt und dort (bezüglich euklidischem Abstand) geclustert. In [PSH+89] werden die nur um den Mittelwert verschobenen Daten in recht direkter Weise verglichen. Zumeist jedoch in Anwendungen dieser Art, so auch in [SMF+98], [SFM+98] und [WMO+97], wird als Ähnlichkeitdefinition der sogenannte Pearson–Korrelationskoeffizient benutzt.

Die Korrelation zweier Zeilen k und l, r_{kl}, schreibt sich dann mit obiger Notation als

$$r_{kl} = \frac{1}{n} \sum_{j=1}^{n} z_{kj} z_{lj} \tag{2.11}$$

Es gilt: $-1 \leq r_{kl} \leq 1$, wobei positive Werte mit steigendem Betrag höhere Korrelation signalisieren und umgekehrt negative Werte auf fehlende Korrelation hinweisen.

Neben bekannten Problemen im Umgang mit diesem Konzept, z.B. der Behandlung fehlender Daten, weisen wir auf zwei wichtige Eigenschaften dieses Korrelationskoeffizienten hin, die in unseren Anwendungen unerwünschte Ergebnisse liefern:

Vergleichen wir zwei Zeilen, deren Einträge bis auf vielleicht sehr wenige Ausnahmen nahezu konstant sind, also eine sehr geringe Standardabweichung aufweisen, so würden wir diese – aus Sicht der Anwendung – gerne als sehr ähnlich betrachten, jedoch gibt der Korrelationskoeffizient dies nicht in jedem Fall wieder.

Als konkretes Beispiel betrachten wir zwei Vektoren $x = (x_1, x_2, \ldots, x_k)$ und $y = (y_1, y_2, \ldots, y_k)$. Sei $x_i = y_i = 1$ für alle $i \in \{1, 2, \ldots, k-2\}$, und $x_{k-1} = 1$, $x_k = 0$, $y_{k-1} = 0$, $y_k = 1$, d.h. die Vektoren unterscheiden sich nur an zwei Stellen, ihr Mittelwert ist $\mu_x = \mu_y = \frac{k-1}{k}$, ihre Standardabweichung $\sigma_x = \sigma_y = \frac{1}{k}\sqrt{k-1}$. Somit ergibt sich für

$$\begin{aligned} r_{xy} &= \frac{1}{k}\Big((k-2)\frac{(1-\frac{k-1}{k})^2}{\frac{k-1}{k^2}}\Big) + 2\frac{(1-\frac{k-1}{k})(-\frac{k-1}{k})}{\frac{k-1}{k^2}} \\ &= \frac{1}{k}\Big(\frac{k-2}{k-1} - 2\Big) \end{aligned}$$

Für $k = 50$ ergibt sich also z.B. $r_{xy} = -0.020408$.

Als Beispiel für eine andere Eigenheit des Pearson–Korrelationskoeffizienten betrachten wir die Vektoren $x = (1, 3, 1, 4, 1, 3, 2, 1, 4, 3, 2, 2, 3)$ und $y = (1, 9, 1, 16, 1, 9, 4, 1, 16, 9, 4, 4, 9)$, es gilt also, daß $y_i = x_i^2 \quad \forall i$. Da wegen $\mu_x = \frac{30}{13}$ und $\mu_y = \frac{84}{13}$ für alle i $x_i - \mu_x$ und $y_i - \mu_y$ das gleiche Vorzeichen haben, erhalten wir mit $r_{xy} = 0.9824$ einen sehr hohen Korrelationskoeffizienten, obwohl der Zusammenhang von x und y quadratisch und nicht linear ist.

2.2.3.1.2 Eine alternative Ähnlichkeitsdefinition Um die eben beschriebenen Szenarien abzumildern und lineare Korrelation im anschaulichen Sinn besser zu fassen, schlagen wir einen alternativen Ansatz vor:

Seien zwei Vektoren $x = (x_1, x_2, \ldots, x_k)$ und $y = (y_1, y_2, \ldots, y_k)$ gegeben. Wir bezeichnen mit $d = (d_1, d_2, \ldots, d_k)$ den Vektor der Distanzen $d_i := x_i - y_i$. Sei m der Median der Werte $\{d_i\}_{i=1}^k$.

Die Ähnlichkeit der beiden Vektoren x und y messen wir nun mit der folgenden Zahl:

$$v_{xy} := \frac{1}{k}\sum_{j=1}^{k} |d_j - m| \tag{2.12}$$

Wir messen also den durchschnittlichen Abstand zweier Komponenten der Vektoren, nachdem wir die Summe dieser Abstände durch Verschiebung um den Median der Distanzen minimiert haben.

Die Ähnlichkeit zweier Vektoren x und y ist also umso höher, je kleiner v_{xy} ist. Eine Normierung der Daten ist nicht notwendig; lineare Korrelation wird sehr gut erkannt und einzelne Ausreißer fallen kaum ins Gewicht, genauso wenig stört eine kleine Standardabweichung, da wir nicht danach normieren.

2.2.3.2 Clusteralgorithmen

Im Verlauf der Arbeit haben wir mehrere verschiedene Kriterien kennengelernt, nach denen Moleküle sinnvoll in Gruppen ähnlicher unterteilt werden können, z.B. Tanimotokoeffizienten, $3D$–Ähnlichkeit und nun Wirkungsähnlichkeit. Meist ist es uns nicht daran gelegen, die gesamte momentan betrachtete Menge von Molekülen komplett zu clustern, sondern wir sind nur daran interessiert, anwendungsbedingt meist kleine Teilmengen von, je nach Maß, sehr ähnlichen Substanzen zu finden.

Zu diesem Zweck eignen sich die oben beschriebenen hierarchischen Verfahren (und werden auch in ähnlichen Zusammenhängen benutzt [SMF$^+$98], [SFM$^+$98], [WMO$^+$97]): Es ist nicht notwendig, zu Beginn des Clusterprozesses eine Anzahl k von Clustern anzugeben. Der Prozess kann an beliebiger Stelle abgebrochen werden, d.h. wir müssen nicht die gesamte Menge unterteilen, sondern finden, aufgrund der Vorgehensweise des hierarchischen Ansatzes, zuerst die Gruppen mit größter Ähnlichkeit. Schließlich erhalten wir mit dem Clusterbaum (Dendogramm) automatisch auch für jedes Cluster noch eine Einteilung in möglicherweise vorhandene interessante Teilcluster.

Das von uns in 3.3.2 benutzte Clusterverfahren ist eine Erweiterung des in 2.2.1 beschriebenen Verfahrens und soll nun hier kurz dargestellt werden.

Wir wollen den Algorithmus in der Sprache der Graphentheorie beschreiben und haben also als Eingabe einen vollständigen, gewichteten Graphen $G = (V, E)$ mit Kantengewichten $w : E \to \mathbb{R}$, wobei die Knoten den Objekten der zu clusternden Menge entsprechen und die Kantengewichte die Abstände zweier Objekte bezeichnen. Wie im Standardverfahren beginnen wir mit der Knotenmenge als der Menge der Cluster und verschmelzen Schritt für Schritt zwei Cluster zu einem neuen. Die Bedingung, die festlegt, welche beiden Cluster verschmolzen werden sollen, unterscheidet unser Verfahren von den bisher benutzten.

Anstatt die Distanz zweier Cluster als Minimum (oder Maximum oder Durchschnitt) der Distanzen der Knoten in den Clustern zu messen, betrachten wir den folgenden Prozess:

1. Sortiere alle Kanten gemäß w absteigend und bezeichne sie mit $e_0, e_1, \ldots, e_{m-1}$.

2. Die Menge $\mathcal{C}$ der Cluster sei gleich der Menge der Knoten von G, $V(G)$.

3. Für jedes Cluster C_i, $0 \leq i \leq n-1$ sei $E_{in}(C_i) = E^j_{out}(C_i) = \emptyset$, $0 \leq j \leq n-1$, $j \neq i$. Setze $i := 0$, $r := n$ und definiere einen Dichtewert $0 \leq \rho \leq 1$ und ein Minimalgewicht w_{min}.

4. Solange $i < m$ und $w(e_i) > w_{min}$: Füge Kante e_i zu $\mathcal{C}$ hinzu.

   (a) Verläuft e_i innerhalb eines Clusters C_j, so setze $E_{in}(C_j) := E_{in}(C_j) \cup \{e_i\}$.

   (b) Verläuft e_i zwischen zwei Clustern C_j und C_k, so aktualisiere $E^k_{out}(C_j) := E^k_{out}(C_j) \cup \{e_i\}$ und $E^j_{out}(C_k) := E^j_{out}(C_k) \cup \{e_i\}$. Falls $|E^k_{out}(C_j)| = |E^j_{out}(C_k)| \geq \rho \cdot |C_k||C_j|$

   - Setze $C_r := C_j \cup C_k$ und $\mathcal{C} := \mathcal{C} \setminus \{C_k, C_j\} \cup \{C_r\}$.
   - Aktualisiere die Mengen $E_{in}(C_r)$ und $E^l_{out}(C_r)$, $E^r_{out}(C_s)$ für alle betroffenen Indizes l und s.
   - Setze $r := r + 1$.

   (c) Setze $i := i + 1$.

Wir können also mit der Wahl des Parameters ρ festlegen, wie dicht ein Cluster zu sein hat, also den Übergang zwischen *single link* und *average link* bzw zwischen *average link* und *complete link* simulieren. Genauso ist es möglich, den Clusterbildungsprozess an einem Punkt abzubrechen, der von der betrachteten Anwendung abhängt ($w(e_i) > w_{min}$). Die in 4b definierte Dichtebedingung ist nur eine Möglichkeit unter ähnlichen anderen.

2.2.4 Biclustern

In diesem Kapitel wollen wir ein neues Verfahren zum Biclustern von Daten vorstellen. Wir präzisieren zuerst die von uns betrachtete Aufgabenstellung: Zu einer Menge O von n Objekten und einer Menge B von m Bedingungen liegt eine Datenmatrix $A \in \mathbb{R}^{n \times m}$ vor, wobei a_{ij} den Meßwert des Objektes i unter der Bedingung j darstellt.

Wir suchen Teilmengen $I \subset O$ von Objekten, die sich jeweils unter einer Teilmenge $J \subset B$ von Bedingungen ähnlich verhalten. Bei der Definition von „ähnlich“ folgen wir [CC00] bzw. [YWWY02]. Dort findet sich eine Bewertungsfunktion, die jedem Cluster einen Wert, das sogenannte Residuum r, zuordnet.

Definition 2.2.1 (Residuum) *Zu einem Cluster C, definiert durch eine Teilmenge I der Objekte und eine Teilmenge J der Bedingungen definieren wir das Residuum r_{IJ} wie folgt: Sei*

$$a_{iJ} := \frac{\sum_{j \in J'} a_{ij}}{|J'|},$$

wobei $J' \subseteq J$ die Teilmenge der Bedingungen darstellt, zu denen für Objekt i tatsächlich Daten vorliegen. Analog sei

$$a_{Ij} := \frac{\sum_{i \in I'} a_{ij}}{|I'|},$$

wobei $I' \subseteq I$ die Teilmenge von Objekten bezeichnet, zu denen unter Bedingung j Daten vorliegen, und schließlich

$$a_{IJ} := \frac{\sum_{i \in I', j \in J'} a_{ij}}{v_{IJ}},$$

wobei mit v_{IJ} das Volumen des Clusters bezeichnet wird, also die tatsächliche Anzahl spezifizierter Einträge.[8]

Nun können wir für jeden Eintrag (i, j) das Residuum

$$r_{ij} := \begin{cases} a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} & \text{, falls } a_{ij} \text{ spezifiziert.} \\ 0 & \text{, sonst.} \end{cases}$$

Schließlich sei das Residuum des Clusters gegeben durch

$$r_{IJ} := \frac{\sum_{i \in I, j \in J} r_{ij}^2}{v_{IJ}}.$$

Die Güte eines Clusters steigt also mit fallendem r, wobei $r = 0$ anzeigt, daß die Zeilen bzw. Spalten der dem Cluster zugeordneten Teilmatrix durch additive Verschiebung auseinander hervorgehen.

Wir benutzen diese Bewertungsfunktion während unseres Verfahrens nicht, sondern nur zum Vergleich der Ergebnisse. Vielmehr zielen wir darauf ab, Teilmatrizen zu identifizieren, welche eine Struktur aufweisen, die automatisch zu Clustern mit guten Residuen führen.

Wir beschreiben im Folgenden den neu entwickelten Algorithmus.

[8] Diese Definition ermöglicht eine direkte Behandlung von Löchern in den Daten.

Aufbau des Graphen G_{ij} Für jedes Paar (i, j) mit $i \in O$ und $j \in B$ konstruieren wir einen bipartiten Graphen G_{ij} auf die folgenden Weise:

1. Für $j, k \in B, k \neq j$ und $i \in O$ definiere $d_{ij}^k := a_{ij} - a_{ik}$.
2. Für festes i und j berechne für alle $k \in B, k \neq j$ die Menge $S_{ij}^k := \{i' \in O : |d_{ij}^k - d_{i'j}^k| \leq t\}$ für ein geeignet gewähltes $t \in \mathbb{R}$.

Anschaulich enthält also die Menge S_{ij}^k all die Objekte, die sich bezüglich der Bedingungen j und k sehr ähnlich zu i verhalten.

Sei nun G_{ij} wie folgt definiert: $V(G_{ij}) = B \cup O$ und $E(G_{ij}) = \{\{b, o\}, b \in B, o \in O : o \in S_{ij}^b\}$.

Ausdünnen des Graphen G_{ij} Zwei weitere Parameter bestimmen nun die Mindestgröße des Clusters, das wir finden wollen: Seien $d_O, d_B \in \mathbb{N}$. Iterativ entfernen wir Knoten aus G_{ij}, um schließlich den Graphen H_{ij} zu erhalten, für den gilt: $V(H_{ij}) \subseteq V(G_{ij})$ und $E(H_{ij}) \subseteq E(G_{ij})$, wobei für alle $o \in O$ gelten soll, daß $deg_{H_{ij}}(o) \geq d_O$, analog für alle $b \in B$: $deg_{H_{ij}}(b) \geq d_B$.

Wir bilden also in gewisser Weise den (d_O, d_B)–Kern des Graphen G_{ij}, indem wir sukzessive die Knoten entfernen, die die Gradbedingungen nicht erfüllen.

Bestimmen dichter Untergraphen von H_{ij} Aufgrund der Konstruktion entsprechen nun gerade die dichten Untergraphen von H_{ij} solchen Teilmatrizen, die ein kleines Residuum haben. Nun bestimmt die Struktur der Eingabedaten, ob bereits die im vorhergehenden Schritt berechneten Kerne ausreichend sind. Falls nicht, so können wir z.B. mit der folgenden Heuristik auch darin noch nach dichteren Teilstrukturen suchen: Für jedes Paar (b, o) von Knoten in H_{ij} betrachten wir den Untergraphen, der induziert wird durch die Nachbarn von o und b. Die dichtesten auf diese Weise entstehenden Untergraphen sind natürliche Kandidaten für gute Cluster. Mit einem weiteren Parameter $0 \leq \alpha \leq 1$ können wir die Mindestdichte kontrollieren.

Wir zeigen in 3.4.2, daß dieser Algorithmus in der Lage ist, Cluster zu finden, die die Ergebnisse von [CC00] und [YWWY02] zum Teil deutlich übertreffen, gemessen nach dem dort definierten Qualitätsmaß, welches wir hier nicht explizit optimieren. Auch bemerken wir, daß die Behandlung von sehr dünnen Daten hier kein Problem darstellt; genauso gilt, daß eindeutige Bicluster, die sich strukturell hervorheben, schnell gefunden werden können. Durch geeignete Wahl der Parameter t, d_O, d_B und α können die Größe, Güte und die Dimensionen der gesuchten Cluster genau justiert werden, wie auch ihre Dichte.

Kapitel 3

Anwendungen

3.1 Clustern molekularer Strukturen

3.1.1 Daten

DIP (The Dictionary of Interfaces in Proteins) [PGF98] ist eine Datenbank, die $3D$–Strukturen von interagierenden Teilen von Proteinen, sogenannten *molecular surface patches (MSPs)*, enthält. Die räumliche Struktur der Oberflächen von Proteinen ist mitverantwortlich für das Zustandekommen von Protein–Ligand– und Protein–Protein–Wechselwirkungen. Aus diesem Grund werden in DIP genau jene Oberflächenregionen, in denen es zu Wechselwirkungen innerhalb des Proteins kommt, genau untersucht. Diese Schnittstellen werden definiert als Paare von MSPs räumlich benachbarter Sekundärstrukturelemente.

DIP versammelt nun eine Vielzahl von solchen MSPs zu einer wachsenden Anzahl von räumlich aufgeklärten Proteinen. Ziel dieser Datensammlung ist es, Vorhersagen zum Docking–Verhalten bestimmter Regionen zu erleichtern und Erkenntnisse über Faltungsprozesse von Proteinen zu gewinnen. Die räumliche Struktur von zwei benachbarten MSPs M_1 und M_2 macht es also möglich, die notwendige Geometrie eines potentiellen Liganden L für eine Oberflächenstruktur O, die räumlich ähnlich ist zu M_1, dadurch vorherzusagen, daß nach Strukturen gesucht wird, die räumlich ähnlich zu M_2 sind [PGF00]. Ein speziell auf die besondere und charakteristische Struktur der MSPs zugeschnittener Überlagerungsalgorithmus, der Teil von DIP ist, macht es möglich, MSPs auf $3D$–Ähnlichkeit zu vergleichen.

Unser Aufgabenstellung im Zusammenhang mit DIP ist die Folgende: Wird ein neues Protein räumlich aufgeklärt, so kann auch dieses in MSPs zerlegt werden. Interessant sind nun die Fragen, ob sich diese neuen MSPs von den schon bekannten in DIP unterscheiden bzw. zu welchen sie sehr

ähnlich sind.

Ein naiver Ansatz besteht darin, ein neues MSP schlicht mit allen in DIP vorliegenden paarweise zu vergleichen. Diese Vorgehensweise wird aber mit zunehmender Größe der Datenbank zu zeitintensiv.

Wir wollen also untersuchen, wie wir die MSPs in DIP organisieren können, so daß zur Suche nach ähnlichen MSPs zu einer neuen Suchstruktur deutlich weniger als alle Elemente von DIP getestet werden müssen.

Aufbereitung der Daten Da der Überlagerungsalgorithmus, der DIP angegliedert ist, nur darauf ausgelegt ist, MSPs von ungefähr gleicher Größe (Atomanzahl) zu vergleichen, haben wir fünf Datensätze von je ca. 1000 MSPs mit jeweis $25-29$, $40-49$, $50-59$, $60-69$ und mehr als 100 Atomen zufällig generiert. Alle MSPs innerhalb einer solchen Menge wurden paarweise auf $3D$–Ähnlichkeit verglichen. Abbildung 3.1 zeigt exemplarisch die Struktur zweier MSPs und deren überlagerte Teilstrukturen.

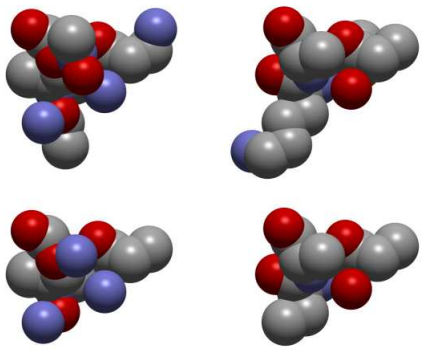


Abbildung 3.1: Zwei MSPs (oben) und ihre überlagerten Teilstrukturen (unten)

3.1.2 Resultate

Die im Folgenden beschriebenen Resultate sind im Wesentlichen schon in [FGG+03] erschienen.

3.1.2.1 Datenanalyse

Da jedes Atom eines MSPs Teil einer Aminosäure des Proteins ist, kann untersucht werden, ob die überlagerten Atome zu zwei MSPs von gleichen Aminosäuren herrühren. Für MSPs mit $30-70$ Atomen handelt es sich meist um

ca. drei bis sieben beteiligte Aminosäuren. Zählen wir die Übereinstimmungen der überlagerten Atome und der Aminosäuren, auf denen diese liegen, so lassen sich grob drei Klassen unterscheiden:

1. Unähnliche Überlagerungen bezüglich dieses Kriterium, d.h. es gibt keine Übereinstimmung auf der Ebene der Aminosäuren.

2. Sequenzähnliche Paare, d.h. solche, die Atome aus mindestens zwei gleichen Aminosäuren überlagern.

3. Sequenzidentische Paare, d.h. die überlagerten Atome stammen immer aus gleichen Aminosäuren.

In Abbildung 3.2 sehen wir exemplarisch die Verteilung der Score–Werte zu einem der Datensätze. Es fällt auf, daß auch hier deutlich drei Regionen unterschieden werden können, die durch die beiden Isoscoren bei 23 und 52 optisch getrennt wurden. Eine statistische Analyse (von durchschnittlichen Score–Werten) zeigte, daß sich die oben aufgeführten drei Klassen von Sequenzähnlichkeiten im Wesentlichen jeweils in den drei optisch erkennbaren Regionen wiederfinden.

Betrachten wir nun die Ergebnisse der Untersuchungen der in 2.2.2.1 definierten Schwellenwertgraphen. Wie dort schon angedeutet, finden wir signifikante strukturelle Unterschiede des Schwellenwertgraphen zu einem zufälligen Graphen mit gleicher Kantenverteilung in Bezug auf die Größe der größten Komponente. Abbildung 3.3 zeigt exemplarisch diese Evolution für einen der betrachteten Datensätze.

Die Kurve, die die Evolution der größten Komponente in G_t beschreibt, zeigt an zwei Stellen einen deutlichen Knick im Vergleich zur Kurve, die denselben Parameter für den zufälligen Graphen H_t beschreibt. Auffallend ist, daß diese beiden Stellen genau mit den Score–Bereichen zusammenfallen, die die oben beschriebenen drei Regionen verschiedener Ähnlichkeit trennen. (Vgl. Abbildung 3.2 mit den beiden Isoscoren zu den Score–Werten 0.23 und 0.52 und Abbildung 3.3 für $t = 23$ und $t = 52$). Diese Übereinstimmung findet sich in ähnlicher Weise für alle fünf untersuchten Datensätze. Die entsprechenden Isoscoren bzw. Knicke verschieben sich abhängig von der Größe der MSPs .

Zusammenfassung Wir haben also gezeigt, daß die aus DIP abgeleiteten Daten reichhaltige Struktur enthalten, die mit zwei unabhängigen Methoden gefunden und als übereinstimmend erkannt werden: Es finden sich im Wesentlichen drei verschiedene Arten von Ähnlichkeit, die einerseits erklärt werden können mit Sequenzähnlichkeit der verglichenen MSPs, andererseits

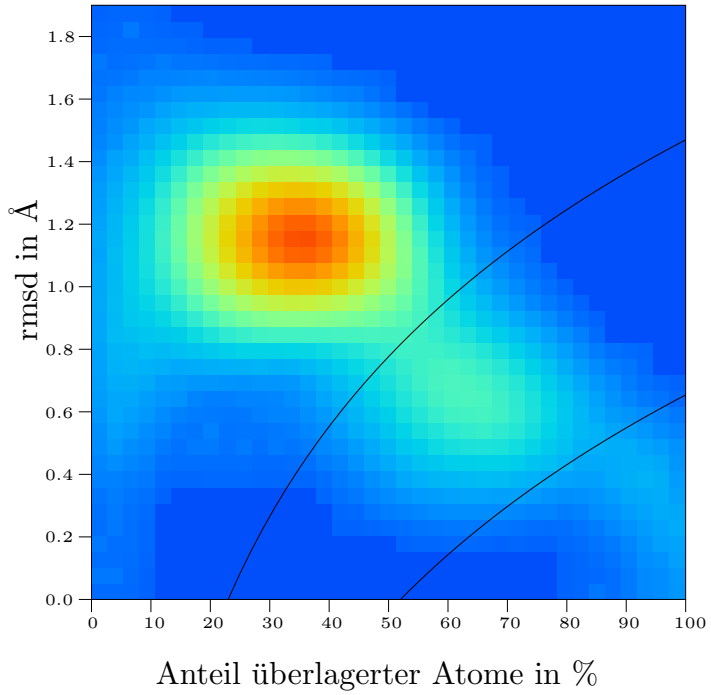


Abbildung 3.2: Verteilung der Scorewerte der paarweisen Vergleiche von 1153 MSPs der Größe ≥ 100

klar zu unterscheiden sind durch Analyse der zugehörigen Schwellenwertgraphen. Zusätzlich gewinnen wir auf diese Weise Informationen darüber, ab welchem Score–Wert von einer signifikanten Ähnlichkeit gesprochen werden kann: genau ab dem Score–Wert, der die Bereiche von sequenzverschiedenen zu sequenzähnlichen MSPs trennt, bzw. in dem Bereich, in dem der erste Knick der Kurve zur Komponentenevolution von G_t auftritt.

3.1.2.2 Sortieren und Suchen

Gerechtfertigt durch die Resultate des vorangegangenen Abschnittes 3.1.2.1 haben wir zwei verschiedene Strategien zum Sortieren der MSPs in DIP entworfen: eine für den Bereich der sehr hohen Score–Werte und eine andere für den Bereich mittlerer Ähnlichkeit. Die Verfahren sind in 2.2.2.2 bzw. 2.2.2.3 beschrieben.

Wir testen die Verfahren in der folgenden Weise: Für jeden der fünf Datensätze werden entsprechend der Suchverfahren die Daten aufbereitet, also

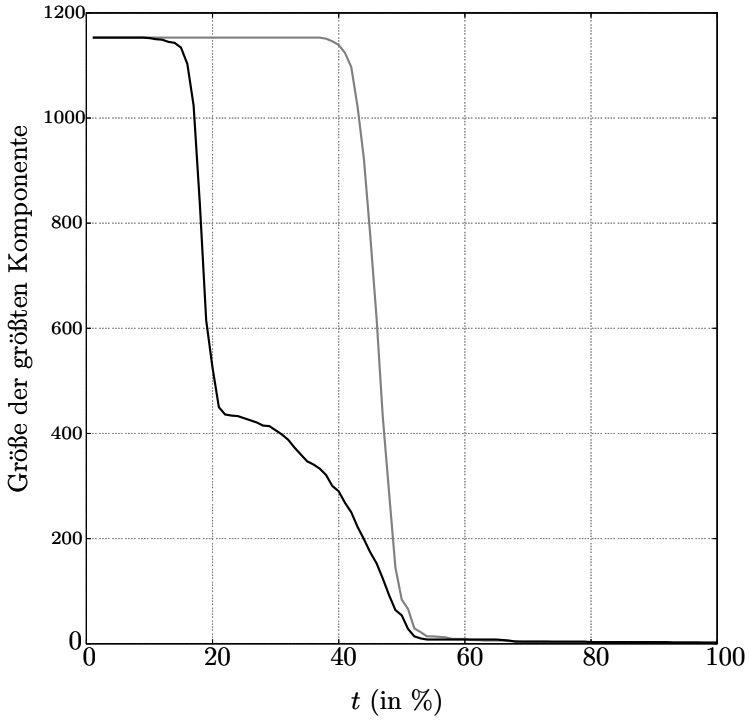


Abbildung 3.3: Evolution der größten Komponente im originalen Schwellenwertgraphen G_t (schwarz) und dem zugehörigen zufälligen Graphen H_t (grau) zu den paarweisen Überlagerungen der 1153 MSPs der Größe ≥ 100 in Abhängigkeit von t.

als binärer Baum für das hierarchische Verfahren (2.2.2.2) bzw. als Repräsentantenstruktur mit Hilfe einer Dominierenden Menge (2.2.2.3). Anschließend spielt jedes MSP des Datensatzes, für welches mindestens ein anderes darin existiert, das eine Ähnlichkeit aufweist, die als signifikant (für den jeweiligen Ähnlichkeitsbereich) betrachtet wird, die Rolle der Suchstruktur N. Wir starten die Suche, speichern die gefundenen Treffer und zählen die Anzahl der benötigten Vergleiche. Schließlich vergleichen wir die Anzahl der gefundenen Treffer und die Anzahl der benötigten Vergleiche mit denen, die sich theoretisch bei einer simplen lineraren Suche ergäben.

Hierarchischer Ansatz Hier werden stets alle zu findenen Treffer, also MSPs, die zu N eine entsprechend hohe Ähnlichkeit haben, gefunden. Die aufgebauten binären Bäume sind fast vollständig balanciert, was eine Reduktion der Anzahl der Fragen zur Folge hat, die der theoretisch besten sehr

nahe kommt. Grund hierfür ist, daß im betrachteten Bereich sehr hoher Ähnlichkeiten starke Transitivität vorliegt, d.h. zwei Strukturen, die eine derart hohe $3D$–Ähnlichkeit haben, sind von fast identischer Struktur, verhalten sich also auch gegenüber dritten fast gleich.

Dominating–Set–Ansatz Wie in 2.2.2.3 beschrieben, steuern die Parameter T und Q die Suche. Sind wir an hohen Trefferquoten interessiert, wählen wir $T = 1$ und $Q = 2\sqrt{|MSPs|}$. Ist das Hauptziel eine möglichst kleine Anzahl von Anfragen, so setzen wir $T = 0$ und $Q = \sqrt{|MSPs|}$. Je nach Datensatz erreichen wir eine Reduktion der Anfragen um eine Faktor von ca. 8 bei einer Trefferquote von ca. 80%. Wollen wir mindestens $85 - 90\%$ aller strukturellen Nachbarn finden, so geht dies immerhin noch ca. dreimal schneller als im naiven Ansatz.

In diesem Ähnlichkeitsbereich können wir auch zeigen, daß je größer die Menge der bekannten MSPs zum Aufbau der Suchstruktur ist, desto schneller und genauer läuft die anschließende Suche.

Hierzu haben wir einen deutlich größeren Datensatz von ca. 5000 MSPs mit je $40-49$ Atomen aufgebaut. Davon wurde jeweils ein Anteil zum Aufbau der Suchstruktur verwendet, der andere bildete die Menge der zu suchenden, neuen Strukturen.

Die in Tabelle 3.1 veranschaulichten Ergebnisse zeigen, daß die Reduktion der Anzahl der Fragen einen Faktor von fast 10 erreicht, bei einer Trefferquote von über 80%. Auch bei sehr hohen Trefferquoten von über 90% wird noch eine ansehnliche Reduktion der Anzahl der Fragen um eine Faktor nahe 5 erreicht.

Zusammenfassung Es konnte gezeigt werden, daß zwei verschiedene, an die Struktur der Daten angepaßte Sortier– und Suchstrategien mit deutlich weniger Anfragen an die Datenbank, verglichen mit der naiven Strategie einer linearen Suche, fast alle relevanten Treffer finden können.

3.1.2.3 Verbesserte Qualität der Überlagerung

Als eine Anwendung des ersten approximativen Überlagerungsalgorithmus aus 2.1.3 haben wir die Elemente der oben erwähnten Menge von ca. 5000 MSPs mit je $40 - 49$ Atomen paarweise auf $3D$–Ähnlichkeit verglichen und die Ergebnisse dieser Berechnungen mit denen verglichen, die der Überlagerungsalgorithmus aus DIP liefert.

Tabelle 3.2 gibt einen Gesamteindruck. Wir sehen an der veränderten Verteilung, daß der neue Überlagerungsalgorithmus insgesamt deutlich höhere Score–Werte berechnet, also bessere Überlagerungen findet. Dies ist vorallem

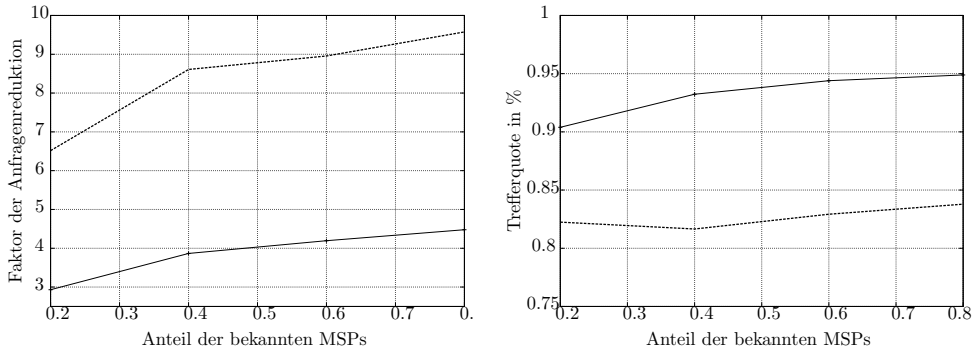


Tabelle 3.1: Reduktion der Anzahl der Anfragen bei wachsender Menge bekannter MSPs (links) und Trefferquote bei wachsender Menge bekannter MSPs (rechts). Die gepunktete Linie entspricht in beiden Fällen der Parameterwahl $T = 0$.

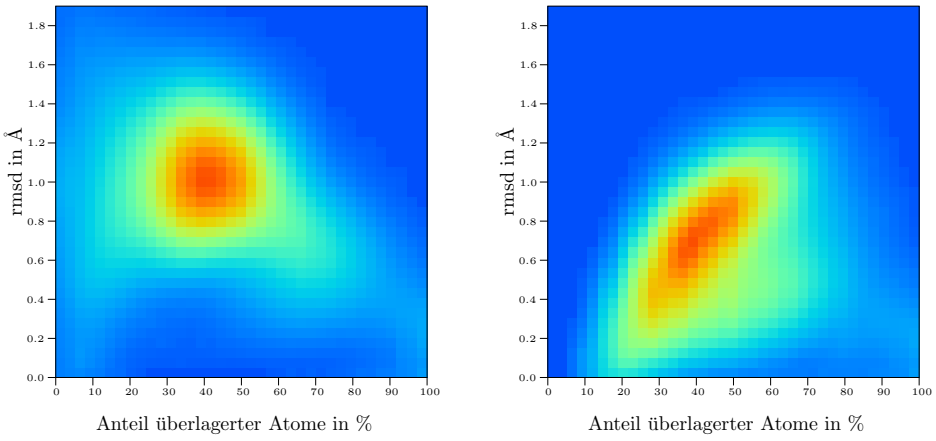


Tabelle 3.2: Verteilung der Scorewerte der paarweisen Vergleiche von 5113 MSPs der Größe $40 - 49$ berechnet mit dem DIP-Überlagerungsalgorithmus (links) und mit dem Überlagerungsalgorithmus aus 2.1.3 (rechts)

im mittleren Ähnlichkeitsbereich der Fall: Die Anzahl der Paare von MSPs mit Score–Werten zwischen 0.33 und 0.75 beträgt für den DIP-Algorithmus 109902, hingegen für unseren neuen Algorithmus 484830. Selbst im Bereich der sehr hohen Ähnlichkeiten ≥ 0.75 erreichen wir noch eine Steigerung von 14% (von 1538 auf 1753 Paare). Allerdings müssen wir auch eine ca. 50–fache Laufzeit aufbringen.

3.2 Ähnlichkeitsvergleich kleiner Moleküle I

In dieser Anwendung zeigen wir, daß der in 2.1.3 beschriebene Überlagerungsalgorithmus in der Lage ist, ähnliche Wirkung bzw. Nebenwirkung von Wirkstoffen zu erkennen.

In einer Datenbank von ca. 2000 Wirkstoffen, dargestellt durch ca. 10^5 Konformere, vergleichen wir 13 Substanzen einer wichtigen Wirkstoffklasse mit dem Rest der Datenbank mit drei verschiedenen Verfahren und vergleichen die Ergebnisse. Es werden jeweils Ähnlichkeiten bezüglich Tanimotokoeffizienten (siehe 2.1.1.2) und $3D$-Ähnlichkeiten berechnet, hier einmal mit einem einfachen, schnellen Überlagerungsalgorithmus (eine Erweiterung des Algorithmus aus DIP (siehe 3.1.1)), einmal mit dem Algorithmus aus 2.1.3. Wir untersuchen, inwieweit die so gefundenen Ähnlichkeiten übereinstimmen mit einer Klasseneinteilung der Wirkstoffe aus klinisch–therapeutischer Sicht (ATC–Codes [WHO02], Anatomical Therapeutic Chemical classification). Es wird gezeigt, daß einerseits teilweise Übereinstimmung besteht zwischen den Ergebnissen der $2D$– und $3D$–Vergleiche, andererseits der Algorithmus aus 2.1.3 geeignet ist, Treffer zu finden, die auf Basis der Tanimotokoeffizienten oder mit dem einfachen Überlagerungsalgorithmus nicht gefunden werden können und somit die benötigte höhere Laufzeit des Verfahrens im Vergleich zu den beiden anderen, für entsprechende Anwendungen gerechtfertigt ist.

Die Ergebnisse dieses Kapitels sind im Wesentlichen in [TGHP04] erschienen.

3.2.1 Daten

ATC–Codes Das ATC–Klassifikationssystems ist ein Hilfsmittel für die Medikamentenforschung und soll helfen, die Qualität der Medikamentenanwendung zu verbessern.

Wirkstoffe werden in verschiedene Gruppen unterteilt, abhängig von den Organen und Systemen, auf die sie wirken, und von ihren chemischen, pharmazeutischen und therapeutischen Eigenschaften.

Die Wirkstoffunterteilung erfolgt in fünf Stufen. Es gibt eine Unterteilung in 14 Hauptklassen (1. Stufe), jeweils mit einer pharmakologisch / therapeutisch Unterklasse (2. Stufe). Die 3. und 4. Stufe sind chemisch / pharmakologisch / therapeutische Untergruppen, die 5. Stufe entspricht der chemischen Substanz.

Für unsere Untersuchung wichtige Untergruppen sind $N05A$ (Antipsychotika) – aus diese Menge werden unsere Suchstrukturen ausgewählt – desweiteren $N06$ (Psychoanaleptika), $R06$ (Antihistaminika zur systemischen Anwendung) und $D04$ (Antipruriginosa, inkl. Antihistaminika, Anästhetika).

Es ist bekannt, daß Substanzen aus der Gruppe $N05$ aufgrund ähnlicher Rezeptoraffinität ähnliche Wirkungen bzw. Nebenwirkungen zeigen können wie Substanzen in den Klassen $N06$, $R06$ und $D04$ [UVM96].

Wirkstoffdatenbank Die betrachtete Datenbank besteht aus ca. 2000 $3D$–Strukturen, wovon jede durch 10-50 Konformere repräsentiert ist (berechnet mit Catalyst [cat] nach dem Verfahren aus [SSHT03]). Eine erweiterte Version dieser Datenbank ist nun öffentlich zugänglich [GDM+05]. Die Datenbank enthält 218 ATC–Klassen ($1. - 3.$ Stufe, wie z.B. $N05A$), von diesen sind 185 mit mindestens drei Substanzen vertreten. Dies entspricht einer Abdeckung von 98% all der Klassen, die mehr als drei Substanzen von kleinem molekularen Gewicht enthalten. Abbildung 3.4 zeigt die Substanzen, die für die Datenbanksuche verwendet wurden.

$2D$–**Vergleich** Die $2D$–Vergleiche wurden mit Hilfe von Tanimotokoeffizienten bezüglich Daylight–Fingerprints [JW95] berechnet.

3.2.2 Resultate

Um die $3D$–Überlagerungen mit dem $2D$–Ansatz zu vergleichen, wurden 13 ausgesuchte Substanzen aus der ATC–Klasse $N05A$ (siehe Abb. 3.4) mit allen Substanzen (mit mehr als 14 Nichtwasserstoffatomen) der Datenbank verglichen, d.h. es wurden jeweils ein Tanimotokoeffizient und ein Score–Wert für die beiden $3D$–Überlagerungsverfahren berechnet. Die Rechenzeit zum Vergleich von zwei einzelnen Konformeren durchschnittlicher Größe (25 Nichtwasserstoffatome) beträgt für den einfachen Überlagerungsalgorithmus ca. 0.01 Sekunden, für den Algorithmus aus 2.1.3 ca. 0.5 Sekunden. Im Folgenden bezeichnen wir den einfachen Überlagerungsalgorithmus mit **A1**, den aus 2.1.3 mit **A2**.

Nun können Tanimotokoeffizienten und $3D$-Scores nicht direkt verglichen werden, d.h. gleiche Zahlenwerte für beide Maße bedeuten nicht gleiche Ähnlichkeit. Für Tanimotokoeffizienten gilt als akzeptiert, daß ab einem Wert von 0.85 von signifikanter Ähnlichkeit gesprochen werden kann. Um einen entsprechenden Wert für den $3D$–Score zu bestimmen, haben wir die Anzahl der Treffer, d.h. Paare von Substanzen, jeweils eine aus der Suchmenge, eine aus der Datenbank, gezählt, die einen Tanimotokoeffizienten ≥ 0.85 aufweisen.

Eine in etwa gleich große Anzahl erhalten wir, wenn wir einen $3D$-Score ≥ 0.75 als signifikant bezeichnen. Genauer erhalten wir 164 Treffer mit Tanimotokoeffizienten ≥ 0.85, **A1** findet 156 Treffer mit $3D$–Score ≥ 0.75, **A2**

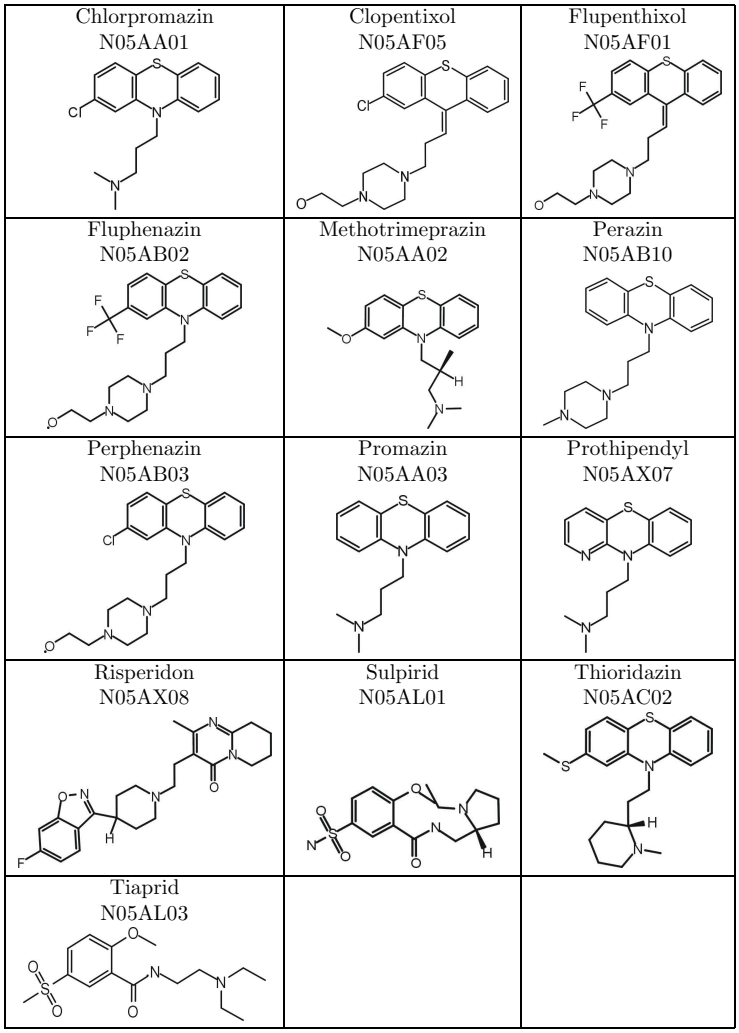


Abbildung 3.4: Verwendete Antipsychotika für die Datenbanksuche: Name, ATC–Code und chemische Struktur

deren 200. Diese Zahlen, aufgeteilt in die verschiedenen ATC-Klassen, finden sich in Tabelle 3.3.

Da wir zeigen wollen, daß der $3D$–Score in der Lage ist, Substanzen mit ähnlicher Wirkung bzw. Nebenwirkung zu erkennen, haben wir zuerst die

	ATC–Code		
Algorithmus	N05A,N06,R06,D04	Andere	$\sum$
Tanimoto > 0.85	113	51	164
A2 > 0.75	157	43	200
A1 > 0.75	131	25	156

Tabelle 3.3: Trefferanzahlen

Treffer in der Datenbank nach ATC–Klassen unterteilt. Für **A1** liegen 84% (131 aus 156) der Treffer in den relevanten Klassen $N05A$, $N06$, $R06$ und $D04$. Für **A2** erhalten wir eine Quote von 78.5% (157 aus 200) und für die $2D$–Ähnlichkeit 69% (113 aus 164)(siehe auch Tabelle 3.3).

Um nun diese Zahlen besser deuten zu können, betrachten wir die folgenden drei verschiedenen Mengen von scheinbar ähnlichen Paaren von Molekülen:

M_1**:** großer Tanimotokoeffizient (≥ 0.85), kleiner $3D$–Score (≤ 0.75)

M_2**:** kleiner Tanimotokoeffizient (≤ 0.85), großer $3D$–Score (≥ 0.75)

M_3**:** großer Tanimotokoeffizient (≥ 0.85), großer $3D$–Score (≥ 0.75)

Betrachten wir die Mengen M_1, M_2 und M_3 nun genauer:

M_1: Für die Substanzen in dieser Menge verhalten sich die Tanimotokoeffizienten zu beiden $3D$-Scores etwa gleich. Jeweils die Hälfte der Treffer in der Datenbank liegt in einer der relevanten ATC–Klassen (siehe Tabelle 3.4 und 3.5 für genaue Zahlen). Eine genauere Untersuchnung dieser Teilmenge (3. Spalte der jeweiligen Tabelle) zeigt, daß in den allermeisten Fällen der entsprechende $3D$–Score einen Wert von zumindest 0.65 aufweist. Wir schließen daraus, da diese Treffer in jedem Fall von biologischer Relevanz sind, daß schon ein $3D$–Score in dieser Größenordnung (≥ 0.65) betrachtenswert ist. Abbildung 3.5a veranschaulicht diesen Sachverhalt.

Die zweite Teilmenge (4. Spalte der jeweiligen Tabelle) mit großen Tanimotokoeffizienten und kleinen $3D$–Scores besteht zumeist aus solchen Treffern, für die der Tanimotokoeffizient die funktionale Ähnlichkeit der Substanzen stark überschätzt. Ein Beispiel findet sich in Abbildung 3.5b.

M_2: Es zeigt sich eine deutlich andere Situation: Die Anzahl der Treffer von **A2** mit hohem $3D$–Score ist wesentlich größer als die von **A1**, sowohl in den relevanten ATC–Klassen, als auch in den anderen (siehe Tabelle 3.4 und 3.5 für exakte Zahlen). Da **A2** keine chemischen Eigenschaften als Eingabe erhält, finden wir einige Treffer, die aus geometrischer Sicht sehr gut sind und

Algorithmus		ATC–Code		
A2	Tanimoto	N05A,N06,R06,D04	Andere	$\sum$
> 0.75	> 0.85	74	11	85
> 0.75	< 0.85	83	32	115
< 0.75	> 0.85	39	40	79

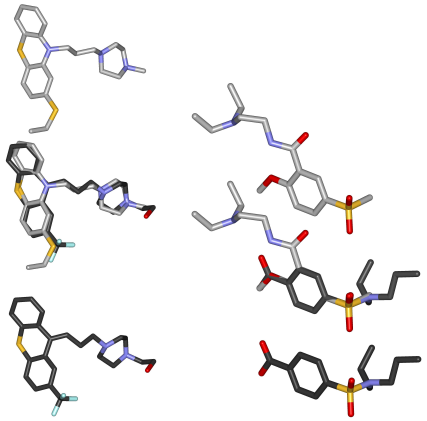
Tabelle 3.4: Trefferanzahl: Tanimoto vs. **A2**

Algorithmus		ATC–Code		
A1	Tanimoto	N05A,N06,R06,D04	Andere	$\sum$
> 0.75	> 0.85	66	10	76
> 0.75	< 0.85	65	15	80
< 0.75	> 0.85	47	41	88

Tabelle 3.5: Trefferanzahl: Tanimoto vs. **A1**

somit einen hohen $3D$–Score erhalten, deren Vorhersagekraft bezüglich ähnlicher Wirkung ist jedoch begrenzt. Diese Treffer finden wir in der 4. Spalte. Die wirklich interessanten Treffer finden sich in der 3. Spalte: Diese sind sowohl aus geometrischer als auch aus biologisch/chemischer Sicht relevant. Ein Grund, warum der $2D$–Ansatz nicht in der Lage ist, diese Treffer zu finden, sind z.B. leichte Veränderungen in der chemischen Struktur. Auch finden wir hier Treffer aus Paaren von Molekülen, die deutlich unterschiedliche Größe haben. Es ist bekannt, daß Tanimotokoeffizienten Ähnlichkeit deutlich verschieden großer Moleküle nur unzureichend erfassen [HSWW03]. Beispiele für die hier angesprochenen Fälle finden sich in Abbildung 3.6. Es läßt sich erkennen, daß gerade für diese Menge von Treffern der Algorithmus **A2** deutlich überlegen ist. Vergleichen wir die Zahlen in Tabelle 3.6 mit den Zahlen in den Tabellen 3.4 und 3.5, so zeigt sich, daß die beiden $3D$–Verfahren sich fast ausschließlich auf genau dieser Treffermenge unterscheiden. Ein wesentlicher Grund dafür ist, daß **A1** nicht in der Lage ist, sehr gute Überlagerungen zu finden, in denen die zu vergleichenden Moleküle von insgesamt recht verschiedener Geometrie sind. Auch deutliche Größenunterschiede der Moleküle, wenn also z.B. eines eine Teilstruktur des anderen ist, können von **A1** kaum erkannt werden. Beispiele dieser Szenarien sind in Abbildung 3.7 zu sehen.

M_3: In dieser Menge finden wir im Wesentlichen die Treffer, die sowohl aufgrund chemischer Struktur als auch in der Größe recht ähnlich sind. Beide Ansätze, $2D$– und $3D$– , finden diese Treffer, und die Resultate sind sehr ähnlich. Auch verhalten sich **A1** und **A2** auf dieser Menge vergleichbar.



(a) (b)

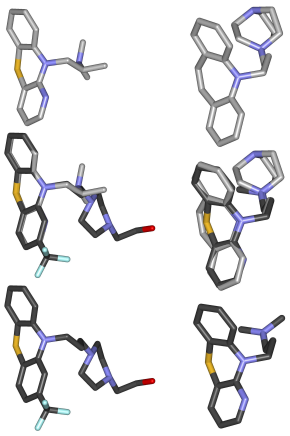
Abbildung 3.5: Vergleich von hoher $2D$–Ähnlichkeit und kleinem $3D$–Score. (a) Überlagerung von Flupentixol (unten, ATC–Code: N05AF01) mit Thiethylperazin (oben, ATC–Code: R06AD03) - mit einem $3D$–Score von 0.63 und einem Tanimotokoeffizienten von 0.89. Die $3D$–Ähnlichkeit wird hier unterbewertet, da die entsprechenden Konformere nicht genau die gleiche Geometrie aufweisen. (b) Überlagerung von Tiaprid (oben, ATC–Code: N05AL03) und Probenecid (unten, ATC–Code: M04AB01) mit einem $3D$–Score von 0.58 und einem Tanimotokoeffizienten von 0.85. Der Tanimotokoeffizient ist hier aufgrund gleicher funktionaler Gruppen, die jedoch simultan nicht überlagerbar sind, wesentlich zu hoch.

Algorithmus		ATC–Code		
A2	**A1**	N05A,N06,R06,D04	Andere	$\sum$
> 0.75	> 0.75	130	22	152
> 0.75	< 0.75	27	21	48
< 0.75	> 0.75	1	3	4

Tabelle 3.6: Trefferanzahl: **A1** vs. **A2**

3.2.2.1 Erhebliche Verbesserung der Laufzeit

Die Analyse der oben geschilderten Resultate war Ausgangspunkt zur Entwicklung des wesentlich schnelleren und grundsätzlich neuen Überlagerungsalgorithmus aus 2.1.4: Bis auf ganz wenige Ausnahmen waren alle Überlagerungen, die **A2** – der Algorithmus aus 2.1.3 – lieferte und die einen hohen Score–Wert besaßen, zusammenhängend (siehe Definition 2.1.8). Diese Tat-



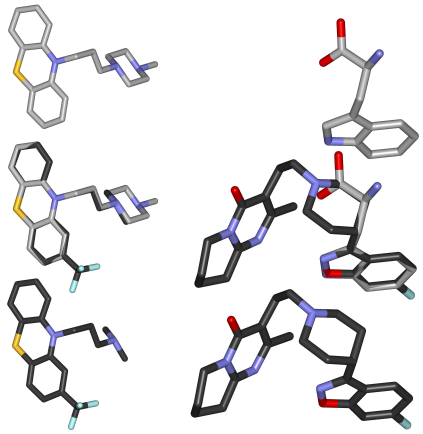
(a) (b)

Abbildung 3.6: Vergleich von hohem $3D$–Score mit kleiner $2D$–Ähnlichkeit: (a) Überlagerung von Fluphenazin (unten, ATC–Code: N05AB02) mit Isothipendyl (oben, ATC–Codes: D04AA22, R06AD09) mit einem $3D$–Score von 0.81 und einem Tanimotokoeffizienten von 0.69. Die Ähnlichkeit zu Isothipendyl (Antihistamin) kann der Tanimotokoeffizient aufgrund fehlender funktionaler Gruppen und der unterschiedlichen Größe der Moleküle nicht erkennen. (b) Überlagerung von Prothipendyl (unten, ATC–Code: N05AX07) und Opipramol (oben, ATC–Code: N06AA05) mit einem $3D$–Score von 0.76 und einem Tanimotokoeffizienten von 0.72. Die Ähnlichkeit zu Opipramol, einem Antidepressivum, wird durch den Tanimotokoeffizienten nicht erkannt, da der mittlere Ring in Opipramol aus sieben Atomen besteht, in Prothipendyl jedoch nur aus sechs Atomen.

sache nutzen wir aus und suchen also im Algorithmus 2.1.4 nur noch nach zusammenhängenden Überlagerungen.

Als ersten Test des Verfahrens aus 2.1.4 wird die obige Datenbanksuche ausgeführt. Die Ergebnisse sind vielversprechend; so ist die Laufzeit erheblich geringer und übertrifft selbst jene von **A1** deutlich. Gemittelt über den vollständigen Datensatz benötigt der Vergleich zweier Konformere nur drei Millisekunden.

Die Qualität der Treffer bleibt erhalten. Von den 2472 Treffern, die **A2** mit einem Score–Wert von mindestens 0.7 findet, verfehlt das Verfahren 2.1.4 nur 69. Die größte Abweichung auf dieser Menge beträgt 4.7%. In 1165 Fällen findet der Algorithmus 2.1.4 bessere Score–Werte als **A2**. Im Bereich der Score–Werte über 0.85 ist 2.1.4 stets mindestens gleichwertig zu **A2**.



(a) (b)

Abbildung 3.7: Unterschiede der beiden $3D$–Verfahren **A1** und **A2**: (a) Überlagerung von Perazin (oben, ATC–Code: N05AD10) mit Triflupromazin (unten, ATC–Code: N05AA05) mit einem $3D$–Score von 0.77 (**A2**), 0.50 (**A1**) und einem Tanimotokoeffizienten von 0.84. Die Ähnlichkeit der beiden Neuroleptika wird von **A1** nicht erkannt, da die Schwerpunkte der Moleküle bei der optimalen Überlagerung nicht aufeinanderfallen. (b) Überlagerung von Tryptophan (oben, ATC–Code: N06AX02) und Risperidon (unten, ATC–Code: N05AX08) mit einem $3D$–Score von 0.77 (**A2**), 0.50 (**A1**) und einem Tanimotokoeffizienten von 0.44. Die Ähnlichkeit zu Tryptophan, einem Antidepressivum, kann **A1** nicht erkennen, da die Gesamtgeometrie zu unterschiedlich zu der von Risperidon ist.

Zusammenfassung Wir haben zuerst gezeigt, daß mit Hilfe von $3D$–Überlagerungsalgorithmen – insbesondere mit **A2**, dem Verfahren aus 2.1.3 – Wirkungsähnlichkeit bzw. ähnliche Nebenwirkungen gut erkannt werden können. Das Verfahren übertrifft insbesondere die herkömmlichen $2D$–Ansätze gerade auf einer Menge von sehr interessanten Treffern, die allein aufgrund ähnlicher Geometrie, jedoch mit unterschiedlichem Aufbau und unterschiedlichen funktionalen Gruppen, Wirkungsähnlichkeit aufzeigen. Das Problem des höheren Laufzeitaufwandes von **A2** im Vergleich zu **A1** und insbesondere zu den $2D$–Methoden ist durch dem Algorithmus 2.1.4 sehr befriedigend gelöst. Große Datenbanken von Molekülen können nun also sehr schnell auf echte $3D$–Ähnlichkeit auf Atomebene durchsucht werden.

3.3 Ähnlichkeitsvergleich kleiner Moleküle II

3.3.1 Daten

Das *Developmental Therapeutics Program (DTP)* des *US National Cancer Institute* hat im *DTP Human Tumor Cell Line Screen* zehntausende von Substanzen auf ihre Fähigkeit getestet, Wachstum von Krebszellen zu mindern. Die Strukturdaten der Substanzen wie auch die Ergebnisse der Experimente sind unter `http://dtp.nci.nih.gov/docs/cancer/cancer_data.html` verfügbar. Der genaue Prozess wie auch allgemeine Informationen über das Programm sind z.B. in [Boy97] und [BP95] zu finden.

Im aktuellen Datensatz (August 2004) liegen experimentelle Ergebnisse (Meßwerte) zu ca. 42000 Substanzen und ca. 60 Krebszellinien vor. Zu jeder Substanz liegt für eine Teilmenge der Zellinien jeweils ein Meßwert vor (der Logarithmus der molaren Konzentration der Substanz, die benötigt wird, um das Wachstum der entsprechenden Krebszelle um 50% zu mindern). Für unsere Untersuchung haben wir diese Daten nach mehreren Kriterien ausgedünnt: Wir wählen zuerst alle Substanzen, zu denen mindestens 10 Konformere vorliegen[1] und die zwischen 10 und 45 Nichtwasserstoffatome besitzen. Aus dieser Menge extrahieren wir jene, die Meßwerte zu mindestens 45 der 60 Zellinien haben, von denen wiederum mindestens zwei Drittel nicht fehlerhaft sind[2]. Im so reduzierten Datensatz befinden sich schließlich ca. 6300 Substanzen. Hieraus haben wir 1200 zufällig ausgewählt. Dies ist der Datensatz, auf den wir uns im Folgenden beziehen.

3.3.2 Resultate

Ziel der Untersuchung ist es zu zeigen, daß $3D$–Ähnlichkeit geeignet ist, Wirkungsähnlichkeit vorherzusagen. Um dies zu belegen, haben wir folgenden Ansatz gewählt: Die 1200 Substanzen des Datensatzes wurden paarweise auf $3D$–Ähnlichkeit verglichen. Zu jeder Substanz S erhalten wir eine nach $3D$–Ähnlichkeit absteigend geordnete Liste L_S^{3D} der zu S ähnlichsten Substanzen. Zum Vergleich legen wir analoge Listen L_S^{2D} für jede Substanz an, die die ähnlichsten Substanzen gemäß Tanimotokoeffizient enthalten.

Auf der anderen Seite clustern wir die Substanzen mit dem Verfahren aus 2.2.3.2 gemäß Wirkungsähnlichkeit; einmal gemessen mit Pearson–Korrelationskoeffizienten (2.2.3.1.1), einmal gemessen mit der alternativen Ähnlichkeitsdefinition aus 2.2.3.1.2. Wir erhalten also zwei Clusterungen C_{pcc} und C_{alt}.

[1] Freundlicherweise bereitgestellt von der Arbeitsgruppe von Robert Preissner.

[2] Es finden sich in den Daten Meßwerte, die offensichtlich fehlerhaft sind.

In der folgenden Weise berechnen wir nun Anreicherungsquoten: Vereinfacht ausgedrückt zählen wir, wie viele der l zu einer Substanz S ähnlichsten Substanzen – gemäß $3D$–Ähnlichkeit – sich in der Clusterung C_{pcc} bzw. C_{alt} gemäß Wirkungsähnlichkeit wiederfinden, wenn wir dort die Cluster betrachten, die die Substanz S enthalten. Diese Anzahl vergleichen wir mit der erwarteten Anzahl $\mathbb{E}(C_S, l)$ von solchen Clustertreffern, hätten wir nicht die Elemente der Liste L_S^{3D} bzw. L_S^{2D} gewählt, sondern l Substanzen zufällig herangezogen. Der genaue Prozess ist in Algorithmus 9 beschrieben.

Algorithmus 9 : Berechnung der Anreicherungsquote

Input : - $\forall$ Substanzen S: Listen L_S der l ähnlichsten Substanzen zu S
- Clusterung $\mathcal{C} := \cup_i C_i$ der Substanzen nach Wirkungsähnlichkeit
- $n_q \leftarrow 0$
- $s_q \leftarrow 0.0$

forall S **do**
  Sei $C_S := \cup C_i$ mit $S \in C_i$ und $|C_i| > 1$
  if $C_S \neq \emptyset$ **then**
    Sei $q_S := \frac{|L_S \cap C_S| - 1}{\mathbb{E}(C_S, l)}$
    $n_q \leftarrow n_q + 1$
    $s_q \leftarrow s_q + q_S$
return $q := \frac{s_q}{n_q}$

Die berechneten Anreicherungsquoten hängen also einerseits ab von den Clusterungen C_{pcc} und C_{alt}, bzw. den Parametern, mit denen diese erzeugt werden (siehe 2.2.3.2). Aufgrund der Verschiedenheiten der beiden Ähnlichkeitsdefinitionen für Wirkungsähnlichkeit müssen wir verschiedene Clusterungsparameter wählen. Für den Pearson–Korrelationskoeffizienten wählen wir ein Mindestgewicht w_{min} von 0.75; den Dichteparameter ρ wählen wir mit 0.8. Für die alternative Ähnlichkeitsdefinition 2.2.3.1.2 wählen wir ein Mindestgewicht, das in etwa gleich viele Kanten für die Clusterung berücksichtigt. Der Dichteparameter kann hier mit 0.2 kleiner gewählt werden.

Bemerkungen Im Allgemeinen zeigen leichte Änderungen dieser Parameter leichte Änderungungen der berechneten Anreicherungsquoten. Jedoch können wir angesichts der nicht allzu hohen Qualität der Daten (siehe 3.3.1) und der in 2.2.3.1.1 schon angesprochenen Schwierigkeiten, mit dem Korrelationskoeffizienten genau die Eigenschaften eines ähnlichen Wirkspektrums zu messen, ohnehin nur tendenzielle Aussagen folgern, ohne die Aussagekraft der Daten und Methoden überzubewerten.

Vergleichs-menge	Anzahl Knoten	Anzahl Cluster	Länge l 1	2	3	5
3D 150	15/200	6/15	160,00	160,00	133,33	133,33
3D 500	37/441	12/15	227,03	178,38	162,16	153,51
3D 1000	49/607	13/15	171,43	110,20	75,51	54,69
3D_2 150	14/201	7/15	257,14	257,14	242,86	242,86
3D_2 200	24/253	9/15	200,00	175,00	158,33	158,33
3D_2 290	27/320	9/15	177,78	133,33	103,70	103,70
3D_2_3 150	22/205	8/15	218,18	190,91	190,91	190,91
3D_2_4 150	25/199	9/15	192,00	168,00	160,00	160,00
3D_2_5 150	22/203	9/15	218,18	245,46	227,27	227,27
2D∩3D 150	14/185	6/15	342,86	385,71	357,14	334,29
2D∩3D 200	21/234	9/15	285,71	257,14	219,05	203,81
2D∩3D 468	33/400	12/15	218,18	182,30	158,06	148,36
2D 150	13/192	7/15	184,62	276,92	246,15	230,77
2D 500	32/385	12/15	243,75	215,62	170,31	150,31
2D 1000	50/562	13/15	168,00	156,32	126,32	111,92

Tabelle 3.7: Anreicherungsquoten: Clusterung nach Wirkung bezüglich alternativer Ähnlichkeitsdefinition 2.2.3.1.2

Der zweite Parameter, der die Anreicherungsquoten verändert, ist die Art, auf die die Listen L_S^* [3] erzeugt werden, wie auch deren Länge l. Wir gehen wie folgt vor: Wie bewerten alle Vergleiche von zwei Substanzen gemäß einem unten beschriebenen Ähnlichkeitsmaß und wählen aus den besten n dieser Vergleiche die Elemente der Listen L_S^*; aus diesen wählen wir dann verschieden lange Anfangsstücke der Länge l. Als Ähnlichkeitsmaße betrachten wir

$3D$**:** $3D$–Ähnlichkeit (Definition 2.1.9)

$3D\_m_1$**,**$3D\_m_1\_m_2$**:** $3D$–Ähnlichkeit gepaart mit topologischen Zusatzinformationen in der folgenden Weise: Wir zählen die zusammenhängenden Komponenten in G_1 und G_2, die von $G(f)$ nicht überdeckt sind. Dies seien k_1 viele. Desweiteren habe die größte dieser Komponenten k_2 Knoten. Wir beziehen also nur solche Überlagerungen mit ein, für die $k_1 \leq m_1$ bzw. $k_1 \leq m_1 \wedge k_2 \leq m_2$ gilt.

$2D$**:** Ähnlichkeit gemäß Tanimotokoeffzient

[3] Der $*$ steht für ein noch zu definierendes Ähnlichkeitsmaß

Vergleichs-	Anzahl	Anzahl	Länge l			
menge	Knoten	Cluster	1	2	3	5
3D 150	39/200	26/61	256,41	256,41	258,08	250,26
3D 500	68/441	45/61	323,53	285,29	269,57	263,23
3D 1000	89/607	52/61	274,16	214,61	189,11	170,56
3D_2 150	39/201	25/61	317,95	333,33	335,00	320,00
3D_2 200	48/253	30/61	358,33	370,83	368,02	355,83
3D_2 290	62/320	38/61	335,48	335,48	320,40	311,61
3D_2_3 150	47/205	27/61	400,00	395,74	391,47	386,38
3D_2_4 150	42/199	24/61	447,62	414,29	414,26	402,38
3D_2_5 150	41/203	24/61	419,51	419,51	425,98	415,61
2D∩3D 150	34/185	22/61	400,00	376,47	366,62	374,71
2D∩3D 200	44/234	28/61	418,18	372,73	356,02	356,36
2D∩3D 468	75/400	44/61	302,93	289,60	282,47	279,47
2D 150	38/192	23/61	368,42	368,42	354,37	349,47
2D 500	61/385	32/61	422,30	363,28	333,74	317,38
2D 1000	84/562	46/61	363,81	316,19	293,13	285,00

Tabelle 3.8: Anreicherungsquoten: Clusterung nach Wirkung bezüglich Pearson–Korrelationskoeffizient 2.2.3.1.1

$2D \cap 3D$**:** gemischte $2D-3D$–Ähnlichkeit in der folgenden Weise: Wir multiplizieren den Tanimotokoeffizienten mit dem $3D$–Score.

Zum Beipiel bezeichne L_S^{3D1000} die Menge der $3D$–ähnlichsten Substanzen von S, die aus der Menge der insgesamt besten 1000 Überlagerungen ausgewählt wurden.

Die auf diese Weise berechneten Anreicherungsquoten finden sich in den Tabellen 3.7 und 3.8. Die erste Spalte gibt an, nach welchem Maß die Vergleiche bewertet werden, und wieviele der besten Vergleiche bezüglich dieses Maßes zu Erstellung der Listen L_S^* herangezogen wurden. In der zweiten Spalte finden wir die Anzahlen der Substanzen, die durch die Mengen der ersten Spalte erhalten wurden (rechter Eintrag), und wieviele davon in den Clustern der entsprechenden Clusterung liegen (linker Eintrag). Die dritte Spalte enthält die Anzahl der Cluster (rechter Eintrag) und wieviele davon durch die Substanzen getroffen werden (linker Eintrag). In den restlichen Spalten finden wir die berechneten Anreicherungsquoten zu den dargestellen Längen l der Listen.

Zusammenfassung In beiden Fällen ist klar sichtbar, daß sowohl $2D$– als auch $3D$–Ähnlichkeit geeignet sind, hohe Anreicherungsquoten zu erreichen.

Auch deutlich wird, daß wir die besten Quoten erreichen, wenn wir jeweils die Listen der ähnlichen Substanzen nur aus der Menge der insgesamt besten Vergleiche bezüglich des gewählten Maßes auswählen, und darin immer nur sehr wenige Elemente betrachten.

Bemerkenswerterweise kann eine rein geometrisch definierte Ähnlichkeit, die keinerlei chemische Information berücksichtigt, ähnlich hohe Anreicherungsquoten erreichen, wie $2D$–Ansätze (siehe Tabelle 3.7). Messen wir die Wirkungsähnlichkeit mit 2.2.3.1.2, so ist ein Maß, welches $2D$– und $3D$–Information gleichzeitig benutzt, den anderen überlegen (siehe Tabelle 3.7). Im Falle von Wirkungsähnlichkeit gemäß Pearson–Korrelationskoeffizienten 2.2.3.1.1 erreichen wir die deutlich besten Quoten mit dem Maß, das neben dem $3D$–Score auch noch berücksichtigt, ob die Überlagerung wesentliche Teile der beiden Moleküle überlagert hat, und nicht nur ein „geometrischer Zufallstreffer“ ist (siehe Tabelle 3.8). Angesichts der sehr guten Laufzeiten liefert der Algorithmus aus 2.1.4 also ein Verfahren, mit dem erstmals wirklich große Datenmengen auf echte $3D$–Ähnlichkeit auf Atomebene durchsucht werden können und dessen Ergebnisse zuverlässig Rückschlüsse auf Wirkungszusammenhänge zulassen, die denen, die mit $2D$–Ansätzen erreicht werden können, z.T. überlegen sind bzw. diese um wertvolle Informationen anreichern können.

3.4 Bicluster

Wir vergleichen in diesem Abschnitt die Ergebnisse unseres Bicluster–Algorithmus aus 2.2.4 mit zwei anderen Verfahren, denen aus [CC00] und [YWWY02]. Vergleiche mit weiteren Ansätzen scheinen wenig sinnvoll, da diese kein mathematisches Kriterium liefern, Bicluster zu bewerten und wir uns einen Vergleich der Ergebnisse aus rein biologisch/medizinischer Sicht nicht anmaßen wollen.

3.4.1 Daten

Drei Datensätze werden in den Arbeiten [CC00] und [YWWY02] betrachtet. Zwei davon enthalten aufbereitete Genexpressionsdaten. Sie sind unter `http://arep.med.harvard.edu/biclustering/` zu finden. Der erste Datensatz, wir nennen ihn im Folgenden **A**, enthält 2884 Objekte unter 17 Bedingungen, der zweite, im Folgenden **B** genannt, 4026 Objekte und 96 Bedingungen. Datensatz **A** ist sehr dicht; es fehlen nur 34 Einträge; in Datensatz **B** fehlen ca. 5% der Einträge. Ein dritter, wir nennen ihn **C**, ist von anderer Gestalt. Der Datensatz beinhaltet Bewertungen von Filmen durch Zuschauer, 943 Zuschauer (Objekte) und 1682 Filme (Bedingungen). Er ist nur sehr dünn besetzt; nur knapp 10% aller Einträge sind vorhanden. Die Daten sind unter `http://www.grouplens.org/` zu erhalten.

3.4.2 Resultate

In [CC00] werden zu den Datensätzen **A** und **B** jeweils 100 Bicluster angegeben. Diese sind in keiner ersichtlichen Weise geordnet, weder nach Größe, also dem Volumen, noch nach Residuum (siehe Definition 2.2.1). Auch finden sich keine Angaben über Mindestdimensionen, also Anzahl von Objekten und Bedingungen, die ein Bicluster mindestens enthalten sollte.

Zur besseren Vergleichbarkeit haben wir die ausgegebenen Bicluster aus [CC00] in etwa nach Residuum geordnet, innerhalb der so enthaltenen Klassen nach Volumen. Die jeweils besten Bicluster, also die größten bei etwa gleichem Residuum, werden zum Vergleich herangezogen.

Zum Vergleich haben wir beim Algorithmus aus 2.2.4 die Parameter in einer Weise eingestellt, daß wir Resultate von jeweils vergleichbarer Größe erhalten. Um Vergleichbarkeit herzustellen, bewerten wir diese anschließend nach ihrem Residuum, welches, wie in 2.2.4 beschrieben, zum Auffinden nicht benutzt wird.

Eine Gegenüberstellung der Resultate findet sich in den Tabellen 3.9 und 3.10, wobei die angegebenen Bicluster des Algorithmus 2.2.4 nur eine kleine

	Bicluster aus [CC00]		Bicluster mit Algorithmus 2.2.4	
Residuum $\leq \cdot$	Volumen	Residuum	Volumen	Residuum
			12016	342.94
≤ 320	11713	318.26	11824	317.47
≤ 300			10175	294.11
≤ 280			10016	263.35
≤ 260			10014	258.77
≤ 240	9515	223.17	9936	224.01
≤ 220	6875	217.58	8696	219.57
≤ 200			7397	196.90
≤ 180			7014	177.62
≤ 160	4945	151.77	5768	149.43
≤ 140			1529	133.25
≤ 120	140	115.56	367	119.21
≤ 115			221	114.97

Tabelle 3.9: Vergleich der Resultate für Datensatz **A**

	Bicluster aus [CC00]		Bicluster mit Algorithmus 2.2.4	
Residuum $\leq \cdot$	Volumen	Residuum	Volumen	Residuum
≤ 900	10468	830.52	11210	867.22
≤ 800			8632	783.57
≤ 750	7566	727.79		
≤ 600			5275	546.96
≤ 500			4420	469.17
			4160	454.96
≤ 400	284	351.98		
≤ 300			1263	298.49
≤ 280			1895	278.79

Tabelle 3.10: Vergleich der Resultate für Datensatz **B**

Bicluster aus [YWWY02]		Bicluster mit Algorithmus 2.2.4	
Volumen	Residuum	Volumen	Residuum
		5134	0.55
		3622	0.51
		3497	0.50
		3338	0.49
		2883	0.47
2755	0.56	2850	0.46
2111	0.51	2700	0.46
1998	0.47	1720	0.41
		513	0.22
		432	0.20
		273	0.15
		132	0.09
		81	0.07

Tabelle 3.11: Vergleich der Resultate für Datensatz **C**

Auswahl darstellen.

Zum Datensatz **C** liegen aus [YWWY02] nur sehr wenig Ergebnisse vor; sie geben nur Volumen und Größe zu drei Biclustern an. Wir stellen diesen unsere Ergebnissen, wiederum nur eine Auswahl, in Tabelle 3.11 gegenüber.

Wir erkennen deutlich, daß der Algorithmus 2.2.4 vorallem für die Datensätze **A** und **C** deutlich bessere Resultate liefert. Diese haben entweder bei etwa gleicher Größe ein deutlich kleineres Residuum, oder aber sind bei vergleichbarem Residuum deutlich größer. Für den Datensatz **B** finden wir Bicluster einer Größe, die in [CC00] nicht gefunden werden.

Zusammenfassung Wir konnten zeigen, daß der Algorithmus 2.2.4 bekannten Ansätzen z.T. deulich überlegen ist. Seine einfache Struktur und die Möglichkeit, über geeignete Parameterwahl die Dimensionen, Dichte und Residuum der gelieferten Bicluster zu Beginn der Suche vorzugeben, machen ihn zu einem geeigneten Suchwerkzeug. Auch konnten wir zeigen, daß sehr dünne Datensätze besser als mit bisherigen Methoden bearbeitet werden können.

Literaturverzeichnis

[Aku93] Akutsu, Tatsuya: A polynomial time algorithm for finding a largest common subgraph of almost trees of bounded degree. In: *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Science* E76-A (1993), S. 1488–1493

[And03] Anderson, Amy C.: The process of structure-based drug design. In: *Chem. Biol.* 10 (2003), S. 787–797

[bab] *Open Babel, http://openbabel.sourceforge.net*

[Baj01] Bajorath, Jürgen: Selected Concepts and Investigations in Compound Classification, Molecular Descriptor Analysis, and Virtual Screening. In: *J. Chem. Inf. Comput. Sci.* 41 (2001), S. 233–245

[Bar93] Barnard, John M.: Substructure Searching Methods: Old and New. In: *J. Chem. Inf. Comput. Sci.* 33 (1993), S. 532–538

[BK73] Bron, C. ; Kerbosch, J.: Finding all cliques of an undirected graph. In: *Commun. ACM* 16 (1973), Nr. 9, S. 575–577

[BKK02] Böhm, Hans-Joachim ; Klebe, Gerhard ; Kubinyi, Hugo: *Wirkstoffdesign.* Heidelberg, Berlin, Oxford : Spektrum Akademischer Verlag, 2002

[Bol01] Bollobás, B.: *Random Graphs.* 2nd edition. Cambridge University Press, 2001

[Boy97] Boyd, M. R.: The NCI In Vitro Anticancer Drug Discovery Screen. Concept, Implementation, and Operation, 1985-1995. In: Teicher, B. (Hrsg.): *Drug Development: Preclinical Screening, Clinical Trials and Approval.* Totowa, NJ : Humana Press, 1997, S. 23–42

[BP95] Boyd, M. R. ; Paull, K. D.: Some Practical Considerations and Applications of the NCI in vitro Drug Discovery Screen. In: *Drug Dev.Res.* 34 (1995), S. 91–109

[cat] *Catalyst*. Accelrys Inc., San Diego, CA. http://www.accelrys.com,

[CC00] Cheng, Yizong ; Church, George M.: Biclustering of expression data. In: *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00)*, 2000, S. 93–103

[CLA80] Carbó, Ramon ; Leyda, Luis ; Arnau, Mariano: How similar is a molecule to another? An electron density measure of similarity between two molecular structures. In: *Int. J. Quantum. Chem.* 17 (1980), S. 1185–1189

[DFK+99] Drineas, P. ; Frieze, A. ; Kannan, R. ; Vempala, S. ; Vinay, V.: Clustering in Large Graphs and Matrices. In: *Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999, S. 291–299

[DNH+92] Dalby, Arthur ; Nourse, James G. ; Hounshell, W. D. ; Gushurst, Ann K. I. ; Grier, David L. ; Leland, Burton A. ; Laufer, John: Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. In: *J. Chem. Inf. Comput. Sci* 32 (1992), S. 244–255

[FFD93] Feuilleaubois, E. ; Fabart, V. ; Doucet, J. P.: Implementation of the three-dimensional-pattern search problem on Hopfield-like neural networks. In: *SAR QSAR Environ Res.* 1 (1993), S. 97–114

[FGG+03] Frömmel, Cornelius ; Gille, Christoph ; Goede, Andrean ; Gröpl, Clemens ; Hougardy, Stefan ; Nierhoff, Till ; Preissner, Robert ; Thimm, Martin: Accelerating screening of 3D protein data with a graph theoretical approach. In: *Bioinformatics* 19 (2003), S. 2442–2447

[Fig72] Figueras, J.: Substructure search by set reduction. In: *J. Chem. Doc.* 12 (1972), S. 237–244

[GDM+05] Goede, Andrean ; Dunkel, Mathias ; Mester, Nina ; Frömmel, Cornelius ; Preissner, Robert: SuperDrug: a conformational drug database. In: *Bioinformatics* 21 (2005), S. 1751–1753

[GHR92] Good, A. C. ; Hodgkin, E. E. ; Richards, W. G.: Utilization of Gaussian functions for the rapid evaluation of molecular similarity. In: *J. Chem. Inf. Comput. Sci.* 32 (1992), S. 188

[Goo85] Goodford, P. J.: A computational procedure for determining energetically favorable binding sites on biologically important macromolecules. In: *J. Med. Chem.* 28 (1985), S. 849–857

[Hås99] Håstad, Johan: Clique is hard to approximate within $n^{1-\epsilon}$. In: *Acta Mathematica* 182 (1999), S. 105–142

[HB97] Hofmann, Thomas ; Buhmann, Joachim M.: Pairwise Data Clustering by Deterministic Annealing. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1997), S. 1–14

[HR87] Hodgkin, E. E. ; Richards, W. G.: Molecular Similarity Based on Electrostatic Potential and Electric Field. In: *Int. J. Quantum Chem. Quantum Biol. Symp.* 14 (1987), S. 105–110

[HRT04] Hougardy, Stefan ; Rocca, Nicolas ; Thimm, Martin: *COSMOS – Comparing Small Molecules for Similarity, preprint.* http://cosmos.informatik.hu-berlin.de, 2004

[HSWW03] Holliday, J. D. ; Salim, N. ; Whittle, M. ; Willett, P.: Analysis and display of the size dependence of chemical similarity coefficients. In: *J. Chem. Inf. Comput. Sci.* 43 (2003), S. 819–828

[HTZ05] Hougardy, Stefan ; Thimm, Martin ; Ziegler, Valentin: *Humboldt Universität zu Berlin: Verfahren und Vorrichtung zum computergestützten Auffinden von ähnlichen Molekülen.* Deutsche Patentanmeldung, Deutsches Patent- und Markenamt, DE 10 2005 029 437.5, 2005

[JMF99] Jain, A. K. ; Murty, M. N. ; Flynn, P. J.: Data clustering: a review. In: *ACM Computing Surveys* 31 (1999), S. 264–323

[JW95] James, C. A. ; Weininger, D.: *Daylight theory manual.* Daylight Chemical Information Systems, Inc.: Irvine, CA, 1995

[JWG95] JONES, G. ; WILLET, P. ; GLEN, R. C.: A genetic algorithm for flexible molecular overlay and pharmacophore elucidation. In: *J. Comput.-Aided Mol. Des.* 9 (1995), S. 532–549

[KD89] KLEIN, R.W. ; DUBES, R. C.: Experiments in projection and clustering by simulated annealing. In: *Pattern Recogn.* 22 (1989), S. 213–220

[KG01] KUROGI, Y. ; GUNER, O. F.: Pharmacophore Modeling and Three-dimensional Database Searching for Drug Design Using Catalyst. In: *Curr. Med. Chem.* 8 (2001), S. 1035–1055

[KHR03] KRÄMER, Andreas ; HORN, Hans W. ; RICE, Julia E.: Fast 3D molecular superposition and similarity search in databases of flexible molecules. In: *J. Comput.-Aided Mol. Des.* 17 (2003), S. 13–38

[Kin67] KING, B.: Step-wise clustering procedures. In: *J. Am. Stat. Assoc.* 69 (1967), S. 86–101

[Kir03] KIRCHNER, Stefan: *Ein Approximationsalgorithmus zur Berechnung der Ähnlichkeit dreidimensionaler Punktmengen*, Humboldt-Universität zu Berlin, Diplomarbeit, 2003

[Kle02] KLEINBERG, Jon: An Impossibility Theorem for Clustering. In: *Advances in Neural Information Processing Systems* 15 (2002)

[Koc01] KOCH, Ina: Fundamental Study: Enumerating all connected maximal common subgraphs in two graphs. In: *Theoretical Computer Science* 250 (2001), S. 1–30

[Koh95] KOHONEN, T.: *Self-Organizing Maps*. Germany : Springer, 1995

[Kub03] KUBINYI, Hugo: QSAR in Drug Design. In: GASTEIGER, Johann (Hrsg.): *Handbook of Chemoinformatics*. Weinheim : Wiley-VCH, 2003, S. 1732–1768

[KVV04] KANNAN, Ravi ; VEMPALA, Santosh ; VETTA, Adrian: On Clusterings: Good, Bad and Spectral. In: *Journal of the ACM* 51 (2004), S. 497–515

[LL97] LEMMEN, Christian ; LENGAUER, Thomas: Time-efficient flexible superposition of medium-sized molecules. In: *J. Comput.-Aided Mol. Des.* 11 (1997), S. 357–368

[LL00] Lemmen, Christian ; Lengauer, Thomas: Computational methods for the structural alignment of molecules. In: *J. Comput.-Aided Mol. Des.* 14 (2000), S. 215–232

[LLK98] Lemmen, Christian ; Lengauer, Thomas ; Klebe, Gerhard: FLEXS: A Method for Fast Flexible Ligand Superposition. In: *J. Med. Chem.* 41 (1998), S. 4502 – 4520

[mac] *MACCS keys*. MDL Information Systems Inc.: San Leandro, CA,

[MBD+93] Martin, Yvonne C. ; Bures, Mark G. ; Danaher, Elizabeth A. ; DeLazzer, Jerry ; Lico, Isabella ; Pavlik, Patricia A.: A fast new approach to pharmacophore mapping and its application to dopaminergic and benzodiazepine agonists. In: *J. Comput.-Aided Mol. Des.* 14 (1993), S. 83–102

[McG82] McGregor, James J.: Backtrack Search Algorithms and the Maximal Common Subgraph Problem. In: *Software Pract. Exper.* 12 (1982), S. 23–34

[McQ67] McQueen, J. B.: Some Methods for Classification and Analysis of Multivariate Observations. In: *Proceedings of the 5th Berkley Symposium on Mathematical Statistics and Probability* I:Statistics (1967), S. 281–297

[MK97] McMahon, Alan J. ; King, Paul M.: Optimization of Carbó molecular similarity index using gradient methods. In: *J. Comput. Chem.* 18 (1997), S. 151–158

[MMM+99] Mason, Jonathan S. ; Morzine, Isabell ; Menard, Paul R. ; Cheney, Daniel L. ; Hulme, Christopher ; Labaudiniere, Richard F.: New 4-Point Pharmacophore Method for Molecular Similarity and Diversity Applications: Overview of the Method and Applications, Including a Novel Approach to the Design of Combinatorial Libraries Containing Privileged Substructures. In: *J. Med. Chem* 42 (1999), S. 3251–3264

[MO04] Madeira, Sara C. ; Oliveira, Arlindo L.: Biclustering Algorithms for Biological Data Analysis: A Survey. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1 (2004), S. 24–45

[NVK+97] Nissink, J. W. M. ; Verdonk, M. L. ; Kroon, J. ; Mietzner, T. ; Klebe, G.: Superposition of molecules: Electron density fitting by application of fourier transforms. In: *J. Comput. Chem.* 18 (1997), S. 638–645

[PGF98] Preissner, Robert ; Goede, Andrean ; Frömmel, Cornelius: Dictionary of interfaces in proteins (DIP). Data bank of complementary molecular surface patches. In: *J. Mol. Biol.* 280 (1998), S. 535–550

[PGF00] Preissner, Robert ; Goede, Andrean ; Frömmel, Cornelius: *Ligandenbestimmung für Proteine.* Deutsche Patentanmeldung, Deutsches Patent- und Markenamt 2000, DE198 31 758 A 1, 2000

[PSH+89] Paull, K. D. ; Shoemaker, R. H. ; Hodes, L. ; Monks, A. ; Scudiero, D. A. ; Rubinstein, L. ; Plowman, J. ; Boyd, M. R.: Display and analysis of patterns of differential activity of drugs against human tumor cell lines: development of mean graph and COMPARE algorithm. In: *Journal of the National Cancer Institute* 81 (1989), S. 1088–1092

[RB79] Raghavan, Vijay V. ; Birchard, Kim: A Clustering Strategy Based on a Formalism of the Reproductive Process in Natural Systems. In: *Proceedings of the Second International Conference on Information Storage and Retrieval*, ACM, 1979, S. 10–22

[RD98] Rarey, Matthias ; Dixon, J. S.: Feature trees: A new molecular similarity measure based on tree matching. In: *J. Comput.-Aided Mol. Des.* 12 (1998), S. 471–490

[RGW02] Raymond, John W. ; Gardiner, Eleonor J. ; Willet, Peter: Heuristics for Similarity Searching of Chemical Graphs Using a Maximum Common Edge Subgraph Algorithm. In: *J. Chem. Inf. Comput. Sci.* 42 (2002), S. 305–316

[RK57] Ray, L. C. ; Kirsch, R. A.: Finding chemical records by digital computers. In: *Science* 126 (1957), S. 814–819

[RSS+02] Rabow, Alfred A. ; Shoemaker, Robert H. ; Sausville, Edward A. ; ; Covell, David G.: Mining the National Cancer Institute's Tumor-Screening Database: Identification of Compounds with Similar Cellular Activities. In: *J. Med. Chem* 45 (2002), S. 818–840

[RW02a] RAYMOND, John W. ; WILLET, Peter: Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2D chemical structure database. In: *J. Comput.-Aided Mol. Des.* 16 (2002), S. 59–71

[RW02b] RAYMOND, John W. ; WILLET, Peter: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. In: *J. Comput.-Aided Mol. Des.* 16 (2002), S. 521–533

[RW03] RAYMOND, John W. ; WILLET, Peter: Similarity Searching in Databases of Flexible 3D Structure Using Smoothed Bounded Distance Geometry. In: *J. Chem. Inf. Comput. Sci.* 43 (2003), S. 908–916

[Sch84] SCHOLLY, A. V.: A relaxation algorithm for generic chemical structure screening. In: *J. Chem. Inf. Comput. Sci.* 24 (1984), S. 235–241

[Sch03] SCHWAB, Christof H.: Conformational Analysis and Searching. In: GASTEIGER, Johann (Hrsg.): *Handbook of Chemoinformatics*. Weinheim : Wiley-VCH, 2003, S. 1732–1768

[SFM+98] SHI, Leming M. ; FAN, Yi ; MYERS, Timothy G. ; O'CONNOR, Patrick M. ; PAULL, Kenneth D. ; FRIEND, Stephen H. ; WEINSTEIN, John N.: Mining the NCI Anticancer Drug Discovery Databases: Genetic Function Approximation for the QSAR Study of Anticancer Ellipticine Analogues. In: *J. Chem. Inf. Comput. Sci.* 38 (1998), S. 189–199

[SMF+98] SHI, Leming M. ; MYERS, Timothy G. ; FAN, Yi ; O'CONNOR, Patrick M. ; PAULL, Kenneth D. ; FRIEND, Stephen H. ; WEINSTEIN, John N.: Mining the National Cancer Institute Anticancer Drug Discovery Database: Cluster Analysis of Ellipticine Analogs with p53-Inverse and Central Nervous System-Selective Patterns of Activity. In: *Mol. Pharmacol.* 53 (1998), S. 241–251

[SSHT03] SMELLIE, A. ; STANTON, R. ; HENNE, R. ; TEIG, S.: Conformational analysis by intersection: CONAN. In: *J. Comput. Chem.* 24 (2003), S. 10–20

[SSK03] SOTRIFFER, Christoph ; STAHL, Martin ; KLEBE, Gerhard: The Docking Problem. In: GASTEIGER, Johann (Hrsg.): *Handbook of Chemoinformatics*. Weinheim : Wiley-VCH, 2003, S. 1732–1768

[STJ+02] STAHL, Martin ; TODOROV, Nikolay P. ; JAMES, Tim ; MAUSER, Harald ; BOEHM, Hans-Joachim ; DEAN, Philip M.: A validation study on the practical use of automated de novo design. In: *J. Comput. Aided. Mol. Des.* 16 (2002), S. 459–478

[Sus65] SUSSENGUTH, E. H.: A graph-theoretic algorithm for matching chemical structures. In: *J. Chem Doc.* 5 (1965), S. 36–43

[Tar77] TARJAN, Robert E.: Graph algorithms in chemical computation. In: CHRISTOFFERSEN, R. E. (Hrsg.): *Algorithms for chemical computations, ACS Symposium Series 46.* Washington DC : American Chemical Society, 1977, S. 1–19

[TGHP04] THIMM, Martin ; GOEDE, Andrean ; HOUGARDY, Stefan ; PREISSNER, Robert: Comparison of 2D Similarity and 3D Superposition. Application to Searching a Coformational Drug Database. In: *J. Chem. Inf. Comput. Sci.* 44 (2004), S. 1816–1822

[TS05] TANAY, Amos ; SHAMIR, Roded Sharan R.: Biclustering Algorithms: A Survey. In: ALURU, Srinivas (Hrsg.): *In Handbook of Computational Molecular Biology.* 1. Chapman & Hall/CRC, 2005. – in Druck

[Ull76] ULLMANN, J. R.: An algorithm for subgraph isomorphism. In: *J. Assoc. Comput. Mach.* 23 (1976), S. 31–42

[Ume91] UMEYAMA, S.: Least Square Estimation of Transformation Parameters Between Two Point Patterns. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13 (1991), S. 376–380

[UVM96] UHL, G. R. ; VANDENBERGH, D. J. ; MINER, L. L.: Knockout mice and dirty drugs. Drug addiction. In: *Curr Biol* 6 (1996), S. 935–936

[WA83] WONG, A. K. C. ; AKINNIYI, F. A.: An algorithm for the largest common subgraph isomorphism using the implicit net. In: *Proc. 1983 Int. Conf. Syst., Man, and Cybern.* (1983), S. 197–201

[WBD98] WILLET, Peter ; BARNARD, John M. ; DOWNS, Geoffrey M.: Chemical Similarity Searching. In: *J. Chem. Inf. Comput. Sci.* 38 (1998), S. 983–996

[Whi32] WHITNEY, H.: Congruent graphs and the connectivity of graphs. In: *Amer. J. Math.* 54 (1932), S. 150–168

[WHO02] WHO: *The selection and use of essential medicines. Report of the WHO Expert Committee (including the 12th Model list of essential 819 medicines).* World Health Organ Tech Rep Ser 2003, 914, i-vi, 1-126, 2002

[WMO+97] WEINSTEIN, John N. ; MYERS, Timothy G. ; O'CONNOR, Patrick M. ; FRIEND, Stephen H. ; FORNACE, Albert J. ; JR. ; KOHN, Kurt W. ; FOJO, Tito ; BATES, Susan E. ; RUBINSTEIN, Lawrence V. ; ANDERSON, N. L. ; BUOLAMWINI, John K. ; OSDOL, William W. ; MONKS, Anne P. ; SCUDIERO, Dominic A. ; SAUSVILLE, Edward A. ; ZAHAREVITZ, Daniel W. ; BUNOW, Barry ; VISWANADHAN, Vellarkad N. ; JOHNSON, George S. ; WITTES, Robert E. ; PAULL, Kenneth D.: An Information-Intensive Approach to the Molecular Pharmacology of Cancer. In: *Science* 275 (1997), S. 343–349

[WSKR01] WALLIS, W. D. ; SHOUBRIDGE, P. ; KRAETZ, M. ; RAY, D.: Graph distances using graph union. In: *Pattern Recogn. Lett.* 22 (2001), S. 701–704

[XB00] XUE, Ling ; BAJORATH, Jürgen: Molecular Descriptors in Chemoinformatics, Computational Combinatorial Chemistry, and Virtual Screening. In: *Comb. Chem. High Throughput Screen.* 3 (2000), S. 363–372

[XGB00] XUE, Ling ; GODDEN, Jeffrey W. ; BAJORATH, Jürgen: Evaluation of descriptors and minifingerprints for the identification of molecules with similar activity. In: *J. Chem. Inf. Comput. Sci.* 40 (2000), S. 1227–1234

[YWWY02] YANG, Jiong ; WANG, Wei ; WANG, Haixun ; YU, Philip: δ-clusters: Capturing subspace correlation in a large data set. In: *Proceedings of the 18th IEEE International Conference on Data Engineering*, 2002, S. 517–528

Danksagung

Ich danke meinem Betreuer Prof. Dr. Hans Jürgen Prömel für die allzeit gute Unterstützung wie auch der DFG für die Förderung im DFG-Forschungszentrum MATHEON.

Meinen Koautoren Prof. Dr. Cornelius Frömmel, Dr. Robert Preissner, Dr. Andrean Goede, Dr. Christoph Gille von der Charité sowie Dr. Till Nierhoff, Dr. Clemens Gröpl und Nicolas Rocca danke ich für die gute Zusammenarbeit, im besonderen Priv.-Doz. Dr. Stefan Hougardy und Valentin Ziegler.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig ohne fremde Hilfe verfaßt und nur die angegebene Literatur und Hilfsmittel verwendet zu haben.
Martin Thimm

25. August 2005